ACM Template

UESTC_Lasagne

Last build at October 16, 2012

Contents

1 To Do List **5**

2 注意事项 **6**

3 字符串处理 **7**

3.1 *AC自动机 . . . 7
3.1.1 指针 . . . 7
3.1.2 非指针 . . . 8
3.2 后缀数组 . . . 9
3.2.1 DC3 . . . 9
3.2.2 DA . . . 11
3.3 后缀三兄弟 . . . 11
3.3.1 例题 . . . 15
3.4 KMP . . . 21
3.5 e-KMP . . . 21
3.6 *Manacher . . . 22
3.7 *字符串最小表示法 . . . 23
3.8 带*通配符的匹配 . . . 23

4 数学 **26**

4.1 扩展GCD . . . 26
4.2 模线性方程组 . . . 26
4.3 矩阵 . . . 27
4.4 康拓展开 . . . 27
4.5 FFT . . . 28
4.6 爬山法计算器 . . . 31
4.7 线性筛 . . . 33
4.8 线性规划 . . . 34
4.9 分解质因数 . . . 37
4.9.1 米勒拉宾+分解因数 . . . 37
4.9.2 暴力版本 . . . 40
4.10 baby step giant step . . . 40
4.11 原根 . . . 41
4.12 逆元 . . . 41
4.13 卢卡斯 . . . 42
4.14 欧拉函数 . . . 42
4.14.1 分解质因数 . . . 42
4.14.2 一次预处理 . . . 42
4.15 费马降阶法 . . . 43
4.16 自适应simp . . . 44
4.17 组合数求模 . . . 44
4.18 其它公式 . . . 46
4.18.1 拉格朗日插值法 . . . 46
4.18.2 正多面体顶点着色 . . . 46
4.18.3 求和公式 . . . 46
4.18.4 几何公式 . . . 47
4.18.5 小公式 . . . 47

5 数据结构 **48**
5.1 *Splay . . . 48
5.2 动态树 . . . 56
5.2.1 维护点权 . . . 56
5.2.2 维护边权 . . . 60
5.3 可持久化线段树 . . . 64
5.4 treap正式版 . . . 67
5.5 树链剖分 . . . 70
5.5.1 点权 . . . 70
5.5.2 边权 . . . 76
5.6 划分树 . . . 80
5.7 树状数组 . . . 81

6 图论 **83**
6.1 优先队列优化的dijkstra . . . 83
6.2 SAP四版 . . . 84
6.3 费用流三版 . . . 86
6.4 匈牙利 . . . 88
6.4.1 新版,隐式图可解 . . . 88
6.4.2 邻接矩阵 . . . 89
6.4.3 邻接表 . . . 89
6.5 一般图匹配带花树 . . . 90
6.6 KM . . . 93
6.6.1 最大加权匹配 . . . 93
6.6.2 自认为正确的Kuhn_Munkras . . . 94
6.7 *二维平面图的最大流 . . . 96
6.8 强联通 . . . 100
6.9 最大团以及相关知识 . . . 101
6.10 双连通分量 . . . 102
6.11 割点与桥 . . . 104
6.12 LCA . . . 106
6.13 最优比例生成树 . . . 108
6.14 全局最小割 . . . 109
6.15 欧拉路 . . . 110
6.15.1 有向图 . . . 110
6.15.2 无向图 . . . 111
6.15.3 混合图 . . . 111
6.16 K短路 . . . 112
6.17 稳定婚姻 . . . 114
6.18 最小树形图 . . . 116

7 计算几何 **119**
7.1 基本函数 . . . 119
7.1.1 Point定义 . . . 119
7.1.2 Line定义 . . . 119
7.1.3 距离：两点距离 . . . 120
7.1.4 距离：点到直线距离 . . . 120
7.1.5 距离：点到线段距离 . . . 120
7.1.6 面积：多边形 . . . 121
7.1.7 判断：线段相交 . . . 121
7.2 圆 . . . 121
7.2.1 面积：两圆相交 . . . 121

7.2.2 三角形外接圆 . . . 122
7.2.3 三角形内切圆 . . . 122
7.2.4 点对圆的两个切点 . . . 122
7.2.5 两圆公切点 . . . 123
7.3 矩阵 . . . 123
7.3.1 基本矩阵 . . . 123
7.3.2 刘汝佳的几何教室 . . . 124
7.4 重心 . . . 127
7.5 KD树 . . . 128
7.5.1 例题 . . . 129
7.6 半平面交 . . . 131
7.7 凸包 . . . 132
7.8 直线与凸包求交点 . . . 133
7.9 三维凸包 . . . 134
7.10 旋转卡壳 . . . 139
7.10.1 单个凸包 . . . 139
7.10.2 两个凸包 . . . 139
7.10.3 外接矩形 . . . 140
7.11 三角形内点个数 . . . 141
7.11.1 无三点共线 . . . 141
7.11.2 有三点共线且点有类别之分 . . . 142
7.12 最近点对 . . . 144
7.12.1 类快排算法 . . . 144
7.12.2 随机增量法 . . . 145
7.13 多圆面积并 . . . 147
7.13.1 去重 . . . 147
7.13.2 圆并 . . . 147
7.14 一个圆与多边形面积交 . . . 150
7.15 精度问题 . . . 151
7.15.1 浮点数为啥会有精度问题 . . . 151
7.15.2 eps . . . 152
7.15.3 eps带来的函数越界 . . . 152
7.15.4 输出陷阱I . . . 152
7.15.5 输出陷阱II . . . 152
7.15.6 范围越界 . . . 152
7.15.7 关于set . . . 153
7.15.8 输入值波动过大 . . . 153
7.15.9 一些建议 . . . 153

8 搜索 **154**
8.1 Dancing Links . . . 154

9 杂物 **157**
9.1 高精度数 . . . 157
9.2 整数外挂 . . . 158
9.3 Java . . . 158
9.3.1 IO . . . 158
9.3.2 优先队列 . . . 158
9.3.3 Map . . . 159
9.3.4 sort . . . 159
9.4 hashmap . . . 159
9.5 C++&STL常用函数 . . . 160

9.5.1 lower_bound/upper_bound 160
9.5.2 rotate 161
9.5.3 nth_element 161
9.5.4 bitset 161
9.5.5 multimap 162
9.6 其它 164
9.6.1 对跑脚本 164
9.6.2 枚举长为n含k个1的01串 164

1 To Do List

所有带*的内容。。。

可以从原来的模板里面继承一些好东西过来。

set,map,multiset等的搞基用法，以及注意事项。

生成树计数

2 注意事项

10^6数量级慎用后缀数组

TLE的时候要冷静哟。。

思考的时候结合具体步骤来的话 会体会到一些不同的东西

C++与G++是很不一样的。。。

map套字符串是很慢的。。。

栈会被记录内存。。。

浮点数最短路要注意取$\le$来判断更新。。。

注意 long long

不要相信.size()

重复利用数组时 小心数组范围

先构思代码框架 每当实际拍马框架变化时 停手 重新思考

有时候四边形不等式也是帮得上忙的 dp 优化是可以水的

结构体里面带数组会非常慢,有时候 BFS 把数组压成数字会快很多。

```
void fun(int a[])
{
    printf("%d\n",sizeof(a));
}
```

结果是 sizeof(a[0]),如果传数组指针然后要清空的话不要用 sizeof。

sqrt 某些时候会出现 sqrt(-0.00)的问题。

将code::blocks的默认终端改成gnome-terminal

```
gnome-terminal -t $TITLE -x
```

最小割割集找法 在残量网络中从源点出发能到的点集记为S原图中S到S'的边即是最小割集

double全局变量初始值可能不是0

3 字符串处理

3.1 *AC自动机

3.1.1 指针

```
const int CHAR=26;
const int TOTLEN=500000;
const int MAXLEN=1000000;
struct Vertex
{
    Vertex *fail,*next[CHAR];
    Vertex(){}
    Vertex(bool flag)//为什么要这样写?
    {
        fail=0;
        memset(next,0,sizeof(next));
    }
};
int size;
Vertex vertex[TOTLEN+1];
void init()
{
    vertex[0]=Vertex(0);
    size=1;
}
void add(Vertex *pos,int cha)
{
    vertex[size]=Vertex(0);
    pos->next[cha]=&vertex[size++];
}
void add(vector<int> s)
{
    int l=s.size();
    Vertex *pos=&vertex[0];
    for (int i=0; i<l; i++)
    {
        if (pos->next[s[i]]==NULL)
            add(pos,s[i]);
        pos=pos->next[s[i]];
    }
}
void bfs()
{
    queue<Vertex *> que;
    Vertex *u=&vertex[0];
    for (int i=0; i<CHAR; i++)
        if (u->next[i]!=NULL)
        {
            que.push(u->next[i]);
            u->next[i]->fail=u;
        }
        else
            u->next[i]=u;
    u->fail=NULL;
    while (!que.empty())
    {
        u=que.front();
        que.pop();
        for (int i=0; i<CHAR; i++)
            if (u->next[i]!=NULL)
            {
                que.push(u->next[i]);
                u->next[i]->fail=u->fail->next[i];
            }
            else
                u->next[i]=u->fail->next[i];
    }
}
```

3.1.2 非指针

```
struct Trie
{
    int next[50][10],fail[50];
    bool end[50];
    int L,root;

    int newNode()
    {
        for (int i = 0;i < 10;i++)
            next[L][i] = -1;
        end[L] = false;
        return L++;
    }

    void Init()
    {
        L = 0;
        root = newNode();
    }

    void Insert(char s[])
    {
        int now = root;
        for (int i = 0;s[i] != 0;i++)
        {
            if (next[now][s[i]-'0'] == -1)
                next[now][s[i]-'0'] = newNode();
            now = next[now][s[i]-'0'];
        }
        end[now] = true;
    }

    void Build()
    {
        queue<int> Q;
        for (int i = 0;i < 10;i++)
            if (next[root][i] == -1)
                next[root][i] = root;
            else
            {
                fail[next[root][i]] = root;
                Q.push(next[root][i]);
            }
        while (!Q.empty())
        {
            int now = Q.front();
            Q.pop();
            end[now] |= end[fail[now]];
            for (int i = 0;i < 10;i++)
                if (next[now][i] == -1)
                    next[now][i] = next[fail[now]][i];
                else
                {
                    fail[next[now][i]] = next[fail[now]][i];
                    Q.push(next[now][i]);
                }
        }
    }
};
```

3.2 后缀数组

3.2.1 DC3

所有下标都是0 n-1，height[0]无意义。

```
//所有相关数组都要开三倍
const int maxn = 300010;
# define F(x) ((x)/3+((x)%3==1?0:tb))
# define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
int wa[maxn * 3], wb[maxn * 3], wv[maxn * 3], ws[maxn * 3];
int c0(int *r, int a, int b)
{
    return r[a] == r[b] && r[a + 1] == r[b + 1] && r[a + 2] == r[
       b + 2];
}
int c12(int k, int *r, int a, int b)
{
    if (k == 2) return r[a] < r[b] || r[a] == r[b] && c12(1, r, a
        + 1, b + 1);
    else return r[a] < r[b] || r[a] == r[b] && wv[a + 1] < wv[b +
        1];
}
void sort(int *r, int *a, int *b, int n, int m)
{
    int i;
    for (i = 0; i < n; i++) wv[i] = r[a[i]];
    for (i = 0; i < m; i++) ws[i] = 0;
    for (i = 0; i < n; i++) ws[wv[i]]++;
    for (i = 1; i < m; i++) ws[i] += ws[i - 1];
    for (i = n - 1; i >= 0; i--) b[--ws[wv[i]]] = a[i];
    return;
}
void dc3(int *r, int *sa, int n, int m)
{
    int i, j, *rn = r + n, *san = sa + n, ta = 0, tb = (n + 1) /
       3, tbc = 0, p;
    r[n] = r[n + 1] = 0;
    for (i = 0; i < n; i++) if (i % 3 != 0) wa[tbc++] = i;
    sort(r + 2, wa, wb, tbc, m);
    sort(r + 1, wb, wa, tbc, m);
    sort(r, wa, wb, tbc, m);
    for (p = 1, rn[F(wb[0])] = 0, i = 1; i < tbc; i++)
        rn[F(wb[i])] = c0(r, wb[i - 1], wb[i]) ? p - 1 : p++;
    if (p < tbc) dc3(rn, san, tbc, p);
    else for (i = 0; i < tbc; i++) san[rn[i]] = i;
    for (i = 0; i < tbc; i++) if (san[i] < tb) wb[ta++] = san[i]
       * 3;
    if (n % 3 == 1) wb[ta++] = n - 1;
    sort(r, wb, wa, ta, m);
    for (i = 0; i < tbc; i++) wv[wb[i] = G(san[i])] = i;
    for (i = 0, j = 0, p = 0; i < ta && j < tbc; p++)
        sa[p] = c12(wb[j] % 3, r, wa[i], wb[j]) ? wa[i++] : wb[j
           ++];
    for (; i < ta; p++) sa[p] = wa[i++];
    for (; j < tbc; p++) sa[p] = wb[j++];
}
//str和sa也要三倍
void da(int str[], int sa[], int rank[], int height[], int n, int
     m)
{
    for (int i = n; i < n * 3; i++)
        str[i] = 0;
    dc3 (str , sa , n + 1 , m);
    int i, j, k;
    for (i = 0; i < n; i++)
    {
        sa[i] = sa[i + 1];
        rank[sa[i]] = i;
    }
    for (i = 0, j = 0, k = 0; i < n; height[rank[i ++]] = k)
        if (rank[i] > 0)
            for (k ? k-- : 0 , j = sa[rank[i] - 1]; i + k < n &&
               j + k < n &&
                    str[i + k] == str[j + k]; k ++) ;
}
```

3.2.2 DA

这份似乎就没啥要注意的了。

```
const int maxn = 200010;
int wx[maxn],wy[maxn],*x,*y,wss[maxn],wv[maxn];

bool cmp(int *r,int n,int a,int b,int l)
{
    return a+l<n && b+l<n && r[a]==r[b]&&r[a+l]==r[b+l];
}
void da(int str[],int sa[],int rank[],int height[],int n,int m)
{
    int *s = str;
    int *x=wx,*y=wy,*t,p;
    int i,j;
    for(i=0; i<m; i++)wss[i]=0;
    for(i=0; i<n; i++)wss[x[i]=s[i]]++;
    for(i=1; i<m; i++)wss[i]+=wss[i-1];
    for(i=n-1; i>=0; i--)sa[--wss[x[i]]]=i;
    for(j=1,p=1; p<n && j<n; j*=2,m=p)
    {
        for(i=n-j,p=0; i<n; i++)y[p++]=i;
        for(i=0; i<n; i++)if(sa[i]-j>=0)y[p++]=sa[i]-j;
        for(i=0; i<n; i++)wv[i]=x[y[i]];
        for(i=0; i<m; i++)wss[i]=0;
        for(i=0; i<n; i++)wss[wv[i]]++;
        for(i=1; i<m; i++)wss[i]+=wss[i-1];
        for(i=n-1; i>=0; i--)sa[--wss[wv[i]]]=y[i];
        for(t=x,x=y,y=t,p=1,i=1,x[sa[0]]=0; i<n; i++)
            x[sa[i]]=cmp(y,n,sa[i-1],sa[i],j)?p-1:p++;
    }
    for(int i=0; i<n; i++) rank[sa[i]]=i;
    for(int i=0,j=0,k=0; i<n; height[rank[i++]]=k)
        if(rank[i]>0)
            for(k?k--:0,j=sa[rank[i]-1]; i+k < n && j+k < n &&
               str[i+k]==str[j+k]; k++);
}
```

3.3 后缀三兄弟

```
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
const int CHAR = 26;
const int MAXN = 100000;
struct SAM_Node
{
    SAM_Node *fa,*next[CHAR];
    int len;
    int id,pos;
    SAM_Node() {}
    SAM_Node(int _len)
    {
        fa = 0;
        len = _len;
        memset(next,0,sizeof(next));
    }
};
SAM_Node SAM_node[MAXN * 2], *SAM_root, *SAM_last;
int SAM_size;
SAM_Node *newSAM_Node(int len)
{
    SAM_node[SAM_size] = SAM_Node(len);
    SAM_node[SAM_size].id=SAM_size;
    return &SAM_node[SAM_size++];
}
SAM_Node *newSAM_Node(SAM_Node *p)
{
    SAM_node[SAM_size] = *p;
    SAM_node[SAM_size].id=SAM_size;
    return &SAM_node[SAM_size++];
}
void SAM_init()
{
    SAM_size = 0;
    SAM_root = SAM_last = newSAM_Node(0);
    SAM_node[0].pos=0;
}
void SAM_add(int x,int len)
{
    SAM_Node *p = SAM_last, *np = newSAM_Node(p->len + 1);
    np->pos=len;
    SAM_last = np;
    for (; p && !p->next[x]; p = p->fa)
        p->next[x] = np;
    if (!p)
    {
        np->fa = SAM_root;
        return ;
    }
    SAM_Node *q = p->next[x];
    if (q->len == p->len + 1)
    {
        np->fa = q;
        return ;
    }
    SAM_Node *nq = newSAM_Node(q);
    nq->len = p->len + 1;
    q->fa = nq;
    np->fa = nq;
    for (; p && p->next[x] == q; p = p->fa)
        p->next[x] = nq;
}
```

```
void SAM_build(char *s)
{
    SAM_init();
    int l = strlen(s);
    for (int i = 0; i < l; i++)
        SAM_add(s[i] - 'a',i+1);
}

SAM_Node * SAM_add(SAM_Node *p, int x, int len)
{
    SAM_Node *np = newSAM_Node(p->len + 1);
    np->pos = len;
    SAM_last = np;
    for (; p && !p->next[x]; p = p->fa)
        p->next[x] = np;
    if (!p)
    {
        np->fa = SAM_root;
        return np;
    }
    SAM_Node *q = p->next[x];
    if (q->len == p->len + 1)
    {
        np->fa = q;
        return np;
    }
    SAM_Node *nq = newSAM_Node(q);
    nq->len = p->len + 1;
    q->fa = nq;
    np->fa = nq;
    for (; p && p->next[x] == q; p = p->fa)
        p->next[x] = nq;
    return np;
}
void SAM_build(char *s)//多串建立 注意SAM_init()的调用
{
    int l = strlen(s);
    SAM_Node *p = SAM_root;
    for (int i = 0; i < l; i++)
    {
        if (!p->next[s[i] - 'a'] || !(p->next[s[i] - 'a']->len ==
            i + 1))
            p=SAM_add(p,s[i] - 'a', i + 1);
        else
            p = p->next[s[i] - 'a'];
    }
}

struct ST_Node
{
    ST_Node *next[CHAR],*fa;
    int len,pos;
```

```
}ST_node[MAXN*2],*ST_root;
int Sufpos[MAXN];
void ST_add(int u,int v,int chr,int len)
{
    ST_node[u].next[chr]=&ST_node[v];
    ST_node[v].len=len;
}
void init(int n)
{
    for (int i=0;i<n;i++)
    {
        ST_node[i].pos=-1;
        ST_node[i].fa=0;
        memset(ST_node[i].next,0,sizeof(ST_node[i].next));
    }
    ST_node[0].pos=0;
    ST_root=&ST_node[0];
}
void ST_build(char *s)
{
    int n=strlen(s);
    reverse(s,s+n);
    SAM_build(s);
    init(SAM_size);
    for (int i=1;i<SAM_size;i++)
    {
        ST_add(SAM_node[i].fa->id,SAM_node[i].id,s[SAM_node[i].
           pos-SAM_node[i].fa->len-1]-'a',SAM_node[i].len-
           SAM_node[i].fa->len);
        if (SAM_node[i].pos==SAM_node[i].len)
        {
            Sufpos[n-SAM_node[i].pos+1]=i;
            ST_node[i].pos=n-SAM_node[i].pos+1;
        }
    }
}

int rank[MAXN],sa[MAXN+1];
int height[MAXN];
int L;
void ST_dfs(ST_Node *p)
{
    if (p->pos!=-1)
        sa[L++]=p->pos;
    for (int i=0;i<CHAR;i++)
        if (p->next[i])
            ST_dfs(p->next[i]);
}
char s[MAXN+1];
int main()
{
    gets(s);
    ST_build(s);
    L=0;
    ST_dfs(ST_root);
    int n=strlen(s);
    for (int i=0; i<n; i++)
        sa[i]=sa[i+1]-1;
    for (int i=0; i<n; i++)
        rank[sa[i]]=i;
    reverse(s,s+n);
    for (int i=0,j=0,k=0; i<n; height[rank[i++]]=k)
        if (rank[i])
            for (k?k--:0,j=sa[rank[i]-1]; s[i+k]==s[j+k]; k++);
}
```

3.3.1 例题

```
#include <iostream>
#include <algorithm>
#include <cstdio>
#include <cstring>
using namespace std;

const int CHAR = 26;
const int MAXN = 100000;

struct SAM_Node
{
    SAM_Node *fa,*next[CHAR];
    int len;
    int id;
    int mat[9];
    SAM_Node() {}
    SAM_Node(int _len)
    {
        fa = 0;
        len = _len;
        memset(mat,0,sizeof(mat));
        memset(next,0,sizeof(next));
    }
};
SAM_Node SAM_node[MAXN*2],*SAM_root,*SAM_last;
int SAM_size;
SAM_Node *newSAM_Node(int len)
{
    SAM_node[SAM_size] = SAM_Node(len);
    SAM_node[SAM_size].id = SAM_size;
    return &SAM_node[SAM_size++];
}
SAM_Node *newSAM_Node(SAM_Node *p)
{
    SAM_node[SAM_size] = *p;
    SAM_node[SAM_size].id = SAM_size;
    return &SAM_node[SAM_size++];
```

```
}
void SAM_init()
{
    SAM_size = 0;
    SAM_root = SAM_last = newSAM_Node(0);
}
void SAM_add(int x,int len)
{
    SAM_Node *p = SAM_last,*np = newSAM_Node(p->len+1);
    SAM_last = np;
    for (; p&&!p->next[x]; p=p->fa)
        p->next[x] = np;
    if (!p)
    {
        np->fa = SAM_root;
        return;
    }
    SAM_Node *q = p->next[x];
    if (q->len == p->len+1)
    {
        np->fa = q;
        return;
    }
    SAM_Node *nq = newSAM_Node(q);
    nq->len = p->len+1;
    q->fa = nq;
    np->fa = nq;
    for (; p&&p->next[x] == q; p = p->fa)
        p->next[x] = nq;
}
int getid(char ch)
{
    return ch-'a';
}
void SAM_build(char *s)
{
    SAM_init();
    int l = strlen(s);
    for (int i = 0; i < l; i++)
        SAM_add(getid(s[i]),i+1);
}
char s[10][MAXN+1];
int ans;
int head[MAXN*2];
struct Edge
{
    int to,next;
} edge[MAXN*2];
int M;
int n;
void add_edge(int u,int v)
{
    edge[M].to=v;
    edge[M].next=head[u];
    head[u]=M++;
}
void dfs(int u)
{
    for (int i=head[u]; i!=-1; i=edge[i].next)
    {
        int v=edge[i].to;
        dfs(v);
        for (int j=0; j<n-1; j++)
            SAM_node[u].mat[j]=max(SAM_node[v].mat[j],SAM_node[u
              ].mat[j]);
    }
    int tmp=SAM_node[u].len;
    for (int i=0; i<n-1; i++)
        tmp=min(tmp,SAM_node[u].mat[i]);
    ans=max(ans,tmp);
}
int main()
{

    while (scanf("%s",s[n])!=EOF)
        n++;
    int L=strlen(s[0]);
    ans=M=0;
    SAM_build(s[0]);
    for (int j=1; j<n; j++)
    {
        int l=strlen(s[j]),len=0;
        SAM_Node *p=SAM_root;
        for (int i=0; i<l; i++)
        {
            if (p->next[getid(s[j][i])])
            {
                p=p->next[getid(s[j][i])];
                p->mat[j-1]=max(p->mat[j-1],++len);
            }
            else
            {
                while (p && !p->next[getid(s[j][i])])
                    p=p->fa;
                if (!p)
                {
                    p=SAM_root;
                    len=0;
                }
                else
                {
                    len=p->len+1;
                    p=p->next[getid(s[j][i])];
                }
```

```
                    p->mat[j-1]=max(p->mat[j-1],len);
                }
            }
    }
    memset(head,-1,4*SAM_size);
    for (int i=1; i<SAM_size; i++)
        add_edge(SAM_node[i].fa->id,i);
    dfs(0);
    printf("%d\n",ans);
    return 0;
}
```

LCS2

```
#include <iostream>
#include <algorithm>
#include <cstdio>
#include <cstring>
using namespace std;

const int CHAR = 26;
const int MAXN = 100000;

struct SAM_Node
{
    SAM_Node *fa,*next[CHAR];
    int len;
    int id;
    int mat[9];
    SAM_Node() {}
    SAM_Node(int _len)
    {
        fa = 0;
        len = _len;
        memset(mat,0,sizeof(mat));
        memset(next,0,sizeof(next));
    }
};
SAM_Node SAM_node[MAXN*2],*SAM_root,*SAM_last;
int SAM_size;
SAM_Node *newSAM_Node(int len)
{
    SAM_node[SAM_size] = SAM_Node(len);
    SAM_node[SAM_size].id = SAM_size;
    return &SAM_node[SAM_size++];
}
SAM_Node *newSAM_Node(SAM_Node *p)
{
    SAM_node[SAM_size] = *p;
    SAM_node[SAM_size].id = SAM_size;
    return &SAM_node[SAM_size++];
}
void SAM_init()
```

```
{
    SAM_size = 0;
    SAM_root = SAM_last = newSAM_Node(0);
}
void SAM_add(int x,int len)
{
    SAM_Node *p = SAM_last,*np = newSAM_Node(p->len+1);
    SAM_last = np;
    for (; p&&!p->next[x]; p=p->fa)
        p->next[x] = np;
    if (!p)
    {
        np->fa = SAM_root;
        return;
    }
    SAM_Node *q = p->next[x];
    if (q->len == p->len+1)
    {
        np->fa = q;
        return;
    }
    SAM_Node *nq = newSAM_Node(q);
    nq->len = p->len+1;
    q->fa = nq;
    np->fa = nq;
    for (; p&&p->next[x] == q; p = p->fa)
        p->next[x] = nq;
}
int getid(char ch)
{
    return ch-'a';
}
void SAM_build(char *s)
{
    SAM_init();
    int l = strlen(s);
    for (int i = 0; i < l; i++)
        SAM_add(getid(s[i]),i+1);
}
char s[MAXN+1];
int ans;
int head[MAXN*2];
struct Edge
{
    int to,next;
} edge[MAXN*2];
int M;
int n;
void add_edge(int u,int v)
{
    edge[M].to=v;
    edge[M].next=head[u];
    head[u]=M++;
}
void dfs(int u)
{
    for (int i=head[u]; i!=-1; i=edge[i].next)
    {
        int v=edge[i].to;
        /*for (int j=0; j<n; j++)
            SAM_node[v].mat[j]=max(SAM_node[v].mat[j],SAM_node[u
               ].mat[j]);*/
        dfs(v);
        for (int j=0; j<n; j++)
            SAM_node[u].mat[j]=max(SAM_node[v].mat[j],SAM_node[u
               ].mat[j]);
    }
    int tmp=SAM_node[u].len;
    for (int i=0; i<n; i++)
        tmp=min(tmp,SAM_node[u].mat[i]);
    ans=max(ans,tmp);
}
int main()
{
    //freopen("in.txt","r",stdin);
    //freopen("out.txt","w",stdout);
    n=0;
    gets(s);
    SAM_build(s);
    while (gets(s))
    {
        int l=strlen(s),len=0;
        SAM_Node *p=SAM_root;
        for (int i=0; i<l; i++)
        {
            if (p->next[getid(s[i])])
            {
                p=p->next[getid(s[i])];
                p->mat[n]=max(p->mat[n],++len);
            }
            else
            {
                while (p && !p->next[getid(s[i])])
                    p=p->fa;
                if (!p)
                {
                    p=SAM_root;
                    len=0;
                }
                else
                {
                    len=p->len+1;
                    p=p->next[getid(s[i])];
                }
                p->mat[n]=max(p->mat[n],len);
            }
            //printf("%d %d %d\n",i,len,p->id);
        }
        n++;
    }
    memset(head,-1,4*SAM_size);
    for (int i=1; i<SAM_size; i++)
        add_edge(SAM_node[i].fa->id,i);
    dfs(0);
    printf("%d\n",ans);
    return 0;
}
```

3.4 KMP

求A[0..i]的一个后缀最多能匹配B的前缀多长。 先对B进行自匹配然后与A匹配。 KMP[i]就是对应答案，p[i]+1是B[0..i]的一个后缀最多能匹配B的前缀多长。

```
//自匹配过程
int j;
p [0] = j = -1;
for ( int i = 1; i < lb; i++)
{
    while (j >= 0 && b[j + 1] != b[i]) j = p[j];
    if (b[j + 1] == b[i]) j ++;
    p[i] = j;
}
//下面是匹配过程
j = -1;
for ( int i = 0; i < la; i++)
{
    while (j >= 0 && b[j + 1] != a[i]) j = p[j];
    if (b[j + 1] == a[i]) j ++;
    KMP[i] = j + 1;
}
```

3.5 e-KMP

求A[i..len-1]和B的最长公共前缀有多长。 先对B进行自匹配然后与A匹配。 eKMP[i]就是对应答案。p[i]是B[i..len-1]和B的最长公共前缀有多长。

```
//自匹配过程
int j = 0;
while (j < lb && b[j] == b[j + 1])
    j++;
p[0] = lb, p[1] = j;
int k = 1;
for (int i = 2; i < lb; i++)
{
    int Len = k + p[k] - 1, L = p[i - k];
    if (L < Len - i + 1)
        p[i] = L;
    else
    {
        j = max(0, Len - i + 1);
        while (i + j < lb && b[i + j] == b[j])
            j++;
        p[i] = j, k = i;
    }
}
//下面是匹配过程
j = 0;
while (j < la && j < lb && a[j] == b[j])
    j++;
eKMP[0] = j;
k = 0;
for (int i = 1; i < la; i++)
{
    int Len = k + eKMP[k] - 1, L = p[i - k];
    if (L < Len - i + 1)
        eKMP[i] = L;
    else
    {
        j = max(0, Len - i + 1);
        while (i + j < la && j < lb && a[i + j] == b[j])
            j++;
        eKMP[i] = j, k = i;
    }
}
```

3.6 *Manacher

待整理

```
char s[1000],a[3000];
int p[3000],len,l,pnow,pid,res,resid;

int main()
{
    while (scanf("%s",s) != EOF)
    {
        len = strlen(s);
        l = 0;
        a[l++] = '.';
        a[l++] = ',';
        for (int i = 0;i < len;i++)
        {
            a[l++] = s[i];
            a[l++] = ',';
        }
        pnow = 0;
        res = 0;
        for (int i = 1;i < l;i++)
        {
            if (pnow > i)
                p[i] = min(p[2*pid-i],pnow-i);
            else
                p[i] = 1;
            for (;a[i-p[i]] == a[i+p[i]];p[i]++);
            if (i+p[i] > pnow)
            {
                pnow = i+p[i];
                pid = i;
            }
            if (p[i] > res)
            {
                res = p[i];
                resid = i;
            }
        }
        for (int i = resid-res+2;i < resid+res-1;i += 2)
            printf("%c",a[i]);
        printf("\n");
    }
    return 0;
}
```

3.7 *字符串最小表示法

```
int Gao(char a[],int len)
{
  int i = 0,j = 1,k = 0;
  while (i < len && j < len && k < len)
  {
    int cmp = a[(j+k)%len]-a[(i+k)%len];
    if (cmp == 0)
      k++;
    else
    {
      if (cmp > 0)
        j += k+1;
      else
        i += k+1;
      if (i == j) j++;
      k = 0;
    }
  }
  return min(i,j);
}
```

3.8 带*通配符的匹配

```
#include <iostream>
#include <algorithm>
#include <cstdio>
#include <cstring>
using namespace std;

```

```
char a[110],b[110],sp[110][110],tot,place[110];
int n,la,lb,ll;

bool check(int id,int pos)
{
    for (int i = 0;sp[id][i] != 0;i++)
        if (b[pos+i] != sp[id][i])
            return false;
    return true;
}

bool check()
{
    lb = strlen(b);
    int pre = 0;
    for (int i = 0;i < tot;i++)
    {
        bool find = false;
        for (int j = pre;j < lb;j++)
            if (check(i,j) == true)
            {
                place[i] = j;
                pre = place[i]+1;
                find = true;
                break;
            }
        if (find == false)  return false;
    }
    if (a[0] != '*')
        if (place[0] != 0)
            return false;
    if (a[la-1] != '*')
        if (check(tot-1,lb-ll) == false)
            return false;
    return true;
}

int main()
{
    while (scanf("%s",a) != EOF)
    {
        tot = 0;
        for (int i = 0;a[i] != 0;i++)
            if (a[i] != '*')
            {
                int j;
                for (j = i;a[j] != 0 && a[j] != '*';j++)
                    sp[tot][j-i] = a[j];
                sp[tot++][j-i] = 0;
                i = j;
            }
        la = strlen(a);
        ll = strlen(sp[tot-1]);
        scanf("%d",&n);
        for (int i = 0;i < n;i++)
        {
            scanf("%s",b);
            if (check() == true)
                puts(b);
        }
    }
    return 0;
}
/*
Sample Input 1
*.*
4
main.c
a.out
readme
yacc

Sample Input 2
*a*a*a
4
aaa
aaaaa
aaaaax
abababa

Sample Output 1
main.c
a.out

Sample Output 2
aaa
aaaaa
abababa
*/
```

4 数学

4.1 扩展GCD

求ax+by=gcd(a,b)的一组解

```
long long ex_gcd(long long a,long long b,long long &x,long long &y)
{
    if (b)
    {
        long long ret = ex_gcd(b,a%b,x,y),tmp = x;
        x = y;
        y = tmp-(a/b)*y;
        return ret;
    }
    else
    {
        x = 1;
        y = 0;
        return a;
    }
}
```

4.2 模线性方程组

```
//有更新
int m[10],a[10];//模数m 余数a
bool solve(int &m0,int &a0,int m,int a)//模线性方程组
{
    int y,x;
    int g=ex_gcd(m0,m,x,y);
    if (abs(a-a0)%g) return 0;
    x*=(a-a0)/g;
    x%=m/g;
    a0=(x*m0+a0);
    m0*=m/g;
    a0%=m0;
    if (a0<0) a0+=m0;
    return 1;
}
int MLES()
{
    bool flag=1;
    int m0=1,a0=0;
    for (int i=0; i<n; i++)
        if (!solve(m0,a0,m[i],a[i]))
        {
            flag=0;
            break;
        }
    if (flag)
```

```
        return a0;
    else
        return -1;
}
```

4.3 矩阵

乘法的时候将B数组转置一下然后$C[i][j] = \sum A[i][k] \times B[j][k]$会有奇效。

```
struct Matrix
{
    int a[52][52];
    Matrix operator * (const Matrix &b)const
    {
        Matrix res;
        for (int i = 0; i < 52; i++)
            for (int j = 0; j < 52; j++)
            {
                res.a[i][j] = 0;
                for (int k = 0; k < 52; k++)
                    res.a[i][j] += a[i][k] * b.a[k][j];
            }
        return res;
    }
    Matrix operator ^ (int y)const
    {
        Matrix res, x;
        for (int i = 0; i < 52; i++)
        {
            for (int j = 0; j < 52; j++)
                res.a[i][j] = 0, x.a[i][j] = a[i][j];
            res.a[i][i] = 1;
        }
        for (; y; y >>= 1, x = x * x)
            if (y & 1)
                res = res * x;
        return res;
    }
};
```

4.4 康拓展开

```
const int PermSize = 12;
int factory[PermSize] = {1, 1, 2, 6, 24, 120, 720, 5040, 40320,
   362880, 3628800, 39916800};
int Cantor(int a[])
{
    int i, j, counted;
    int result = 0;
    for (i = 0; i < PermSize; ++i)
    {
        counted = 0;
        for (j = i + 1; j < PermSize; ++j)
```

```
            if (a[i] > a[j])
                ++counted;
        result = result + counted * factory[PermSize - i - 1];
    }
    return result;
}

bool h[13];

void UnCantor(int x, int res[])
{
    int i,j,l,t;
    for (i = 1;i <= 12;i++)
        h[i] = false;
    for (i = 1; i <= 12; i++)
    {
        t = x / factory[12 - i];
        x -= t * factory[12 - i];
        for (j = 1, l = 0; l <= t; j++)
            if (!h[j])l++;
        j--;
        h[j] = true;
        res[i - 1] = j;
    }
}
```

4.5 FFT

```
const double PI= acos(-1.0);
struct vir
{
  double re,im; //实部和虚部
  vir(double a=0,double b=0)
  {
    re=a;
    im=b;
  }
  vir operator +(const vir &b)
  {return vir(re+b.re,im+b.im);}
  vir operator -(const vir &b)
  {return vir(re-b.re, im-b.im);}
  vir operator *(const vir &b)
  {return vir(re*b.re-im*b.im , re*b.im+im*b.re);}
};
vir x1[200005],x2[200005];
void change(vir *x,int len,int loglen)
{
  int i,j,k,t;
  for(i=0;i<len;i++)
  {
    t=i;
    for(j=k=0; j<loglen; j++,t>>=1)
      k= (k<<1)|(t&1);
    if(k<i)
    {
    //  printf("%d %d\n",k,i);
      vir wt=x[k];
      x[k]=x[i];
      x[i]=wt;
    }
  }
}
void fft(vir *x,int len,int loglen)
{
  int i,j,t,s,e;
  change(x,len,loglen);
  t=1;
  for(i=0;i<loglen;i++,t<<=1)
  {
    s=0;
    e=s+t;
    while(s<len)
    {
      vir a,b,wo(cos(PI/t),sin(PI/t)),wn(1,0);
      for(j=s;j<s+t;j++)
      {
        a=x[j];
        b=x[j+t]*wn;
        x[j]=a+b;
        x[j+t]=a-b;
        wn=wn*wo;
      }
      s=e+t;
      e=s+t;
    }
  }
}
void dit_fft(vir *x,int len,int loglen)
{
  int i,j,s,e,t=1<<loglen;
  for(i=0;i<loglen;i++)
  {
    t>>=1;
    s=0;
    e=s+t;
    while(s<len)
    {
      vir a,b,wn(1,0),wo(cos(PI/t),-sin(PI/t));
      for(j=s;j<s+t;j++)
      {
        a=x[j]+x[j+t];
        b=(x[j]-x[j+t])*wn;
        x[j]=a;
        x[j+t]=b;
        wn=wn*wo;
      }
      s=e+t;
      e=s+t;
    }
  }
  change(x,len,loglen);
  for(i=0;i<len;i++)
    x[i].re/=len;
}
int main()
{
  char a[100005],b[100005];
  int i,len1,len2,len,loglen;
  int t,over;
  while(scanf("%s%s",a,b)!=EOF)
  {
    len1=strlen(a)<<1;
    len2=strlen(b)<<1;
    len=1;loglen=0;
    while(len<len1)
    {
      len<<=1;  loglen++;
    }
    while(len<len2)
    {
      len<<=1;  loglen++;
    }
    for(i=0;a[i];i++)
    {
      x1[i].re=a[i]-'0';
      x1[i].im=0;
    }
    for(;i<len;i++)
      x1[i].re=x1[i].im=0;
    for(i=0;b[i];i++)
    {
      x2[i].re=b[i]-'0';
      x2[i].im=0;
    }
    for(;i<len;i++)
      x2[i].re=x2[i].im=0;
    fft(x1,len,loglen);
    fft(x2,len,loglen);
    for(i=0;i<len;i++)
      x1[i] = x1[i]*x2[i];
    dit_fft(x1,len,loglen);
    for(i=(len1+len2)/2-2,over=len=0;i>=0;i--)
    {
      t=(int)(x1[i].re+over+0.5);
      a[len++]= t%10;
      over = t/10;
    }
    while(over)
    {
      a[len++]=over%10;
      over/=10;
    }
    for(len--;len>=0&&!a[len];len--);
      if(len<0)
      putchar('0');
      else
        for(;len>=0;len--)
          putchar(a[len]+'0');
    putchar('\n');
  }
  return 0;
}
```

4.6 爬山法计算器

注意灵活运用。
双目运算符在calc()中，左结合单目运算符在P()中，右结合单目运算符在calc_exp中。（但是还没遇到过。。）

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <string>
using namespace std;

char s[100000];
int n,cur;
const string OP = "+-*";

char next_char()
{
    if (cur >= n) return EOF;
    return s[cur];
}

int get_priority(char ch)
{
    if (ch == '*')  return 2;
    return 1;
}

int P();

int calc(int a,char op,int b)
{
    if (op == '+')
        return a+b;
    if (op == '-')
        return a-b;
    if (op == '*')
        return a*b;
}

int calc_exp(int p)
{
    int a = P();
    while ((OP.find(next_char()) != OP.npos) && (get_priority(
       next_char()) >= p))
    {
        char op = next_char();
        cur++;
        a = calc(a,op,calc_exp(get_priority(op)+1));
    }
    return a;
}

int totvar,m,var[26],varid[26];

int P()
{
    if (next_char() == '-')
    {
        cur++;
        return -P();
    }
    else if (next_char() == '+')
    {
        cur++;
        return P();
    }
    else if (next_char() == '(')
    {
        cur++;
        int res = calc_exp(0);
        cur++;
        return res;
    }
    else
    {
        cur++;
        //cout << "getvar at " << cur << ' ' << var[varid[s[cur
           ]-'a']] << endl;
        return var[varid[s[cur-1]-'a']];
    }
}

int id[26],minid;

int main()
{
    while (true)
    {
        scanf("%d%d",&totvar,&var[0]);
        if (totvar == 0 && var[0] == 0)  break;
        for (int i = 1;i < totvar;i++)
            scanf("%d",&var[i]);
        scanf("%d",&m);
        scanf("%s",s);
        for (int i = 0;i < 26;i++)
            id[i] = -1;
        minid = 0;
        n = strlen(s);
        for (int i = 0;i < n;i++)
            if (s[i] >= 'a' && s[i] <= 'z')
            {
                if (id[s[i]-'a'] == -1)
                {
                    id[s[i]-'a'] = minid;
                    minid++;
                }
                s[i] = 'a'+id[s[i]-'a'];
            }
        for (int i = 0;i < totvar;i++)
            varid[i] = i;
        int res = 0;
        do
        {
            cur = 0;
            int tmp = calc_exp(0);
            if (tmp == m)
            {
                res++;
                break;
            }
        }
        while (next_permutation(varid,varid+totvar));
        //puts(s);
        if (res > 0)
            puts("YES");
        else
            puts("NO");
    }
  return 0;
}
```

4.7 线性筛

```
int N;
bool isPrime[10001];
int prime[10000];
void getPrime(int n)
{
    memset(isPrime,1,++n);
    N=0;
    isPrime[0]=isPrime[1]=0;
    for (int i=2;i<n;i++)
    {
        if (isPrime[i])
            prime[N++]=i;
        for (int j=0;j<N && prime[j]*i<n;j++)
        {
            isPrime[i*prime[j]]=0;
            if (i%prime[j]==0)
                break;
        }
    }
}
```

4.8 线性规划

```
#define MAXM 20  //max num of basic varibles
#define INF 1E200

double A[MAXM+5][MAXN+MAXM+5];
double b[MAXM+5],c[MAXN+MAXM+5];
int N[MAXN+5],B[MAXM+5];
double X[MAXN+MAXM+5],V;
int n,m,R,C,nCnt,bCnt;
int v1[MAXN],v2[MAXN];

int fcmp(double a,double b)
{
  if(fabs(a-b)<1E-7) return 0;
  if(a>b) return 1;
  return -1;
}

void Pivot(int l,int e)
{
  double t=A[l][e],p=c[e];
  b[l]=b[l]/t;
  for(int i=1;i<=C;i++)
    A[l][i]/=t;
  V=V-c[e]*b[l];
  for(int i=1;i<=R;i++)
  {
    if(i==l||fcmp(A[i][e],0.0)==0)
      continue;
    t=A[i][e];
    b[i]=b[i]-t*b[l];
    for(int j=1;j<=C;j++)
      A[i][j]=A[i][j]-t*A[l][j];
  }
  for(int i=1;i<=C;i++)
    c[i]=c[i]-p*A[l][i];
  for(int i=1;i<=nCnt;i++)
  {
    if(N[i]==e)
    {
      N[i]=B[l];
      break;
    }
  }
  B[l]=e;
}

bool Process(double P[])
{
  while(true)
  {
    int e=-1;
    double mV=-INF;
    for(int i=1;i<=nCnt;i++)
      if(fcmp(P[N[i]],mV)==1)
        mV=P[N[i]],e=N[i];

    if(fcmp(mV,0.0)<=0) break;
    int l=-1;
    mV=INF;
    for(int i=1;i<=bCnt;i++)
    {
      if(fcmp(A[i][e],0.0)==1)
      {
        double t=b[i]/A[i][e];
        if(fcmp(mV,t)==1||(fcmp(mV,t)==0&&(l==-1||B[l]>B[i])))
          mV=t,l=i;
      }
    }
    if(l==-1) return false;
    Pivot(l,e);
  }
  return true;
}

bool initSimplex()
{
  nCnt=bCnt=0;
  for(int i=1;i<=n;i++)
    N[++nCnt]=i;
  for(int i=1;i<=m;i++)
    B[++bCnt]=i+n,A[i][n+i]=1.0;
  R=bCnt,C=bCnt+nCnt;
  double minV=INF;
  int p=-1;
  for(int i=1;i<=m;i++)
    if(fcmp(minV,b[i])==1)
      minV=b[i],p=i;
  if(fcmp(minV,0.0)>=0)
```

```
    return true;
  N[++nCnt]=n+m+1;R++,C++;
  for(int i=0;i<=C;i++)
    A[R][i]=0.0;
  for(int i=1;i<=R;i++)
    A[i][n+m+1]=-1.0;
  Pivot(p,n+m+1);
  if(!Process(A[R])) return false;
  if(fcmp(b[R],0.0)!=0)
    return false;
  p=-1;
  for(int i=1;i<=bCnt&&p==-1;i++)
    if(B[i]==n+m+1) p=i;
  if(p!=-1)
  {
    for(int i=1;i<=nCnt;i++)
    {
      if(fcmp(A[p][N[i]],0.0)!=0)
      {
        Pivot(p,N[i]);
        break;
      }
    }
  }
  bool f=false;
  for(int i=1;i<=nCnt;i++)
  {
    if(N[i]==n+m+1) f=true;
    if(f&&i+1<=nCnt)
      N[i]=N[i+1];
  }
  nCnt--;
  R--,C--;
  return true;
}

//-1: no solution 1: no bound 0: has a solution -V
int Simplex()
{
  if(!initSimplex())
    return -1;
  if(!Process(c))
    return 1;
  for(int i=1;i<=nCnt;i++)
    X[N[i]]=0.0;
  for(int i=1;i<=bCnt;i++)
    X[B[i]]=b[i];
  return 0;
}

int main()
{
    //n = 1;m=1;
    //V= 0.0;
    //c[1] = 1.0;
    //A[1][1] = 1.0;
    //b[1] = 5.0;
    //Simplex();
    //printf("V = %.3f\n",V);

  while(scanf("%d",&v1[1]) == 1)
        {
            for(int i = 2; i<=6;i++)
                scanf("%d",&v1[i]);
            n = 4; m = 6;
            for(int i = 0 ; i<=m+1;i++)
                for(int j=0;j<=n+m+2;j++)
                    A[i][j] = c[j] = 0;
            memset(b,0,sizeof(b));
            V = 0.0;
            /*
            n 为未知数个数
            m 为约束个数
            目标: siama(c[i]*xi)
            约束: sigma(A[i][j]*xj) <=b[i]; j = 1 ...  n
            解存在X里面
            */
            b[1] = v1[1] ; A[1][1] = 1;A[1][4] = 1;
            b[2] = v1[2] ; A[2][1] = 1;A[2][3] = 1;
            b[3] = v1[3] ; A[3][3] = 1;A[3][4] = 1;
            b[4] = v1[4] ; A[4][2] = 1;A[4][3] = 1;
            b[5] = v1[5] ; A[5][2] = 1;A[5][4] = 1;
            b[6] = v1[6] ; A[6][1] = 1;A[6][2] = 1;
            c[1] = 1;c[2] = 1;c[3] = 1;c[4] = 1;
            Simplex();
            //printf("V = %.3f\n",V);
            printf("%.3f␣%.3f␣%.3f␣%.3f\n",X[1],X[2],X[3],X[4]);

        }
  return 0;
}
```

4.9 分解质因数

4.9.1 米勒拉宾+分解因数

```
#include<ctime>
#include<iostream>
#define bint long long
using namespace std;
const int TIME = 8;//测试次数, 够了8~10
int factor[100],fac_top = -1;

//计算两个数的gcd
bint gcd(bint small,bint big)
```

```
{
    while(small)
    {
        swap(small,big);
        small%=big;
    }
    return abs(big);
}

//ret = (a*b)%n (n<2^62)
bint muti_mod(bint a,bint b,bint n)
{
    bint exp = a%n, res = 0;
    while(b)
    {
        if(b&1)
        {
            res += exp;
            if(res>n) res -= n;
        }
        exp <<= 1;
        if (exp>n) exp -= n;
        b>>=1;
    }
    return res;
}

// ret = (a^b)%n
bint mod_exp(bint a,bint p,bint m)
{
    bint exp=a%m, res=1; //
    while(p>1)
    {
        if(p&1)
            res=muti_mod(res,exp,m);
        exp = muti_mod(exp,exp,m);
        p>>=1;
    }
    return muti_mod(res,exp,m);
}

//miller-法测试素数rabin, time 测试次数
bool miller_rabin(bint n, int times)
{
    if(n==2)return 1;
    if(n<2||!(n&1))return 0;
    bint a, u=n-1, x, y;
    int t=0;
    while(u%2==0)
    {
        t++;
        u/=2;
    }
    srand(time(0));
    for(int i=0; i<times; i++)
    {
        a = rand() % (n-1) + 1;
        x = mod_exp(a, u, n);
        for(int j=0; j<t; j++)
        {
            y = muti_mod(x, x, n);
            if ( y == 1 && x != 1 && x != n-1 )
                return false; //must not
            x = y;
        }
        if( y!=1) return false;
    }
    return true;
}

bint pollard_rho(bint n,int c)//找出一个因子
{
    bint x,y,d,i = 1,k = 2;
    srand(time(0));
    x = rand()%(n-1)+1;
    y = x;
    while(true)
    {
        i++;
        x = (muti_mod(x,x,n) + c) % n;
        d = gcd(y-x, n);
        if (1 < d && d < n) return d;
        if( y == x) return n;
        if(i == k)
        {
            y = x;
            k <<= 1;
        }
    }
}

void findFactor(bint n,int k)//二分找出所有质因子，存入factor
{
    if(n==1)return;
    if(miller_rabin(n, TIME))
    {
        factor[++fac_top] = n;
        return;
    }
    bint p = n;
    while(p >= n)
        p = pollard_rho(p,k--);//值变化，防止死循环k
    findFactor(p,k);
    findFactor(n/p,k);
```

```
}

int main()
{
    bint cs,n,min;
    cin>>cs;
    while (cs--)
    {
        cin>>n;
        fac_top = min = -1;
        if(miller_rabin(n,TIME)) cout<<"Prime"<<endl;
        else
        {
            findFactor(n,107);
            for(int i=0; i<=fac_top; i++)
            {
                if(min<0||factor[i]<min)
                    min = factor[i];
            }
            cout<<min<<endl;
        }
    }
    return 0;
}
```

4.9.2 暴力版本

```
int N;
int num[30],fac[30];
void getFactor(int x)
{
    N=0;
    memset(num,0,sizeof(num));
    for (int i=0; prime[i]*prime[i]<=x && i<L; i++)
    {
        if (x%prime[i]==0)
        {
            while (x%prime[i]==0)
            {
                x/=prime[i];
                num[N]++;
            }
            fac[N++]=prime[i];
        }
    }
    if (x>1)
    {
        num[N]=1;
        fac[N++]=x;
    }
}
```

4.10 baby step giant step

```
#define MOD 76543
int hs[MOD], head[MOD], next[MOD], id[MOD], top;
void insert(int x, int y)
{
    int k = x%MOD;
    hs[top] = x, id[top] = y, next[top] = head[k], head[k] = top
       ++;
}
int find(int x)
{
    int k = x%MOD;
    for (int i = head[k]; i; i = next[i]) if (hs[i] == x) return
       id[i];
    return -1;
}
int BSGS(int a, int b, int n)
{
    memset(head, 0, sizeof(head));
    top = 1;
    if (b==1) return 0;
    int m = sqrt(n+.0), j;
    long long x = 1, p = 1;
    for (int i = 0; i < m; ++i, p = p*a%n) insert(p*b%n, i);
    for (long long i = m; ; i += m)
    {
        if ((j = find(x=x*p%n)) != -1) return i-j;
        if (i > n) break;
    }
    return -1;
}
```

4.11 原根

```
int getPriRoot(int p)
{
    if (p==2) return 1;
    int phi = p - 1;
    getFactor(phi);
    for (int g = 2; g < p; ++g)
    {
        bool flag=1;
        for (int i = 0; flag && i < N; ++i)
            if (power(g, phi/fac[i], p) == 1)
                flag=0;
        if (flag)
            return g;
    }
}
```

4.12 逆元

```
void getInv2(int x)
{
    inv[1]=1;
    for (int i=2; i<=x; i++)
        inv[i]=(mod-(mod/i)*inv[mod%i]%mod)%mod;
}
int getInv(int x)//为素数mod
{
    return power(x,mod-2);
}
```

4.13 卢卡斯

卢卡斯，$num[i]$阶乘也

```
int comLucus(int n,int m,int p)
{
    int ans=1;
    for (; n && m && ans; n/=p,m/=p)
    {
        if (n%p>=m%p)
            ans = ans*num[n%p]%p*getInv(num[m%p]%p)%p*getInv(num[
                n%p-m%p])%p;
        else
            ans=0;
    }
    return ans;
}
```

4.14 欧拉函数

4.14.1 分解质因数

```
int getEuler(int x)
{
    getFactor(x);
    int ret=x;
    for (int i=0; i<N; i++)
        ret = ret/fac[i]*(fac[i]-1);
    return ret;
}
```

4.14.2 一次预处理

```
void getEuler2()
{
    memset(euler,0,sizeof(euler));
    euler[1] = 1;
    for (int i = 2; i <= 3000000; i++)
    {
        if (!euler[i])
        {
            for (int j = i; j <= 3000000; j += i)
            {
                if (!euler[j])
                    euler[j] = j;
                euler[j] = euler[j]/i*(i-1);
            }
        }
    }
}
```

4.15 费马降阶法

分解素数p为x^2+y^2的费马降阶法，失败返回-1，主程序调用calcu(p,x,y)

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int p,expp,A,B,aa,ans,tt;
long long M;
long long exp(int a,int b,long long mod)
{
     long long ans=1,num=a;
     while (b!=0)
     {
           if (b&1)
           {
                   ans=((ans%mod)*(num%mod))%mod;
           }
           num=((num%mod)*(num%mod))%mod;
           b>>=1;
     }
     return ans;
}
int calcu(int p,int &x,int &y)
{
          if (p%4!=1) return -1;
          else
          {
              expp=(p-1)/4;
              A,B;
              while (1)
              {
                  aa=rand()%p;
                  if (aa==0) continue;
                  A=exp(aa,expp,p);
                  ans=(((long long)A%p)*((long long)A%p))%p;
                  if (ans==p-1) break;
              }
              B=1;
              M=((long long)A*(long long)A+(long long)B*(long
                 long)B)/p;
              if (M!=1) B=p;
              while (M!=1)
              {
                   if (B>A)
                   {tt=A; A=B; B=tt;}
                   tt=A;
                   A=B;
```

```
                    B=tt%B;
                    M=((long long)A*(long long)A+(long long)B*(long long)B)/p;
              }
              if (B<=A)
              {
                       x=B;
                       y=A;
              }
              else
              {
                 x=A;
                 y=B;
              }
          }
}
int main()
{
    while (scanf("%d",&p)!=EOF)
    {
          int x,y;
          if (calcu(p,x,y)!=-1)
    }
    return 0;
}
```

4.16 自适应simp

过了哈尔滨积分题，精度要求不高的时候可以考虑使用。
暂时我只能用这个做做类似于凸函数或者凹函数的函数。

```
double Simp(double l,double r)
{
    double h = (r-l)/2.0;
    return h*(calc(l)+4*calc((l+r)/2.0)+calc(r))/3.0;
}

double rSimp(double l,double r)
{
    double mid = (l+r)/2.0;
    if (abs((Simp(l,r)-Simp(l,mid)-Simp(mid,r)))/15 < eps)
        return Simp(l,r);
    else
        return rSimp(l,mid)+rSimp(mid,r);
}
```

4.17 组合数求模

模是质数

```
#include<cstdio>
#include<cstring>
#include<iostream>
```

```
using namespace std;
int mod;
long long num[100000];
int ni[100],mi[100];
int len;
void init(int p)
{
    mod=p;
    num[0]=1;
    for (int i=1; i<p; i++)
        num[i]=i*num[i-1]%p;
}
void get(int n,int ni[],int p)
{
    for (int i = 0; i < 100; i++)
        ni[i] = 0;
    int tlen = 0;
    while (n != 0)
    {
        ni[tlen++] = n%p;
        n /= p;
    }
    len = tlen;
}
long long power(long long x,long long y)
{
    long long ret=1;
    for (long long a=x%mod; y; y>>=1,a=a*a%mod)
        if (y&1)
            ret=ret*a%mod;
    return ret;
}
long long getInv(long long x)//mod为素数
{
    return power(x,mod-2);
}
long long calc(int n,int m,int p)//C(n,m)%p
{
    init(p);
    long long ans=1;
    for (; n && m && ans; n/=p,m/=p)
    {
        if (n%p>=m%p)
            ans = ans*num[n%p]%p*getInv(num[m%p]%p)%p*getInv(num[
               n%p-m%p])%p;
        else
            ans=0;
    }
    return ans;
}
int main()
{
    int t;
    scanf("%d",&t);
    while (t--)
    {
        int n,m,p;
        scanf("%d%d%d",&n,&m,&p);
        printf("%I64d\n",calc(n+m,m,p));
    }
    return 0;
}
```

4.18 其它公式

4.18.1 拉格朗日插值法

已知$y = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1}$曲线上的$n$个点$(x_1, y_1), (x_2, y_2), (x_3, y_3) \cdots (x_n, y_n)$用拉格朗日插值法可以不求系数可知任意$x$对应的$y$值。

$$\begin{aligned} y = {} & y_1\frac{(x-x_2)(x-x_3)\cdots(x-x_n)}{(x_1-x_2)(x_1-x_3)\cdots(x_1-x_n)} \\ & + y_2\frac{(x-x_1)(x-x_3)\cdots(x-x_n)}{(x_2-x_1)(x_2-x_3)\cdots(x_2-x_n)} \\ & + \cdots \\ & + y_n\frac{(x-x_1)(x-x_2)\cdots(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)\cdots(x_n-x_{n-1})} \end{aligned}$$

特别的，如果$x_1 \sim x_n$为 连续自然数，那么对于下一个自然数对应的y值为：

$$y_{n+1} = (-1)^{n-1}C_n^0y_1 + (-1)^{n-2}C_n^1y_2 + \cdots + (-1)^0C_n^{n-1}y_n$$

这个组合系数可以通过高斯消元暴出来，前提是要猜到它满足递推关系。

4.18.2 正多面体顶点着色

正四面体：$N = \frac{(n^4+11\times n^2)}{12}$
正六面体：$N = \frac{(n^8+17\times n^4+6\times n^2)}{24}$
正八面体：$N = \frac{(n^6+3\times n^4+12\times n^3+8\times n^2)}{24}$
正十二面体：$N = \frac{(n^{20}+15\times n^{10}+20\times n^8+24\times n^4)}{60}$
正二十面体：$N = \frac{(n^{12}+15\times n^6+44\times n^4)}{60}$

4.18.3 求和公式

$\sum k = \frac{n\times(n+1)}{2}$
$\sum 2k-1 = n^2$
$\sum k^2 = \frac{n\times(n+1)\times(2n+1)}{6}$
$\sum (2k-1)^2 = \frac{n\times(4n^2-1)}{3}$
$\sum k^3 = (\frac{n\times(n+1)}{2})^2$
$\sum (2k-1)^3 = n^2\times(2n^2-1)$
$\sum k^4 = \frac{n\times(n+1)\times(2n+1)\times(3n^2+3n-1)}{30}$
$\sum k^5 = \frac{n^2\times(n+1)^2\times(2n^2+2n-1)}{12}$

$\sum k \times (k+1) = \frac{n\times(n+1)\times(n+2)}{3}$
$\sum k \times (k+1) \times (k+2) = \frac{n\times(n+1)\times(n+2)\times(n+3)}{4}$
$\sum k \times (k+1) \times (k+2) \times (k+3) = \frac{n\times(n+1)\times(n+2)\times(n+3)\times(n+4)}{5}$

4.18.4 几何公式

球扇形：
全面积：$T = \pi r(2h + r_0)$，h为球冠高，r_0为球冠底面半径
体积：$V = \frac{2\pi r^2 h}{3}$

4.18.5 小公式

Pick 公式：$A = E \times 0.5 + I - 1$（A是多边形面积，E是边界上的整点，I是多边形内部的整点）

海伦公式：$S = \sqrt{p(p-a)(p-b)(p-c)}$，其中$p = \frac{(a+b+c)}{2}$，$abc$为三角形的三条边长

求$\binom{n}{k}$中素因子P的个数：

1. 把n转化为P进制，并记它每个位上的和为$S1$
2. 把$n-k$，k做同样的处理，得到$S2$，$S3$

则$\binom{n}{k}$中素因子P的个数：$\frac{S2+S3-S1}{P-1}$

部分错排公式：
$n+m$个数中m个数必须错排 求排列数

```
dp[i] = n*dp[i-1]+(i-1)*(dp[i-1]+dp[i-2]);
dp[0] = n!;
dp[1] = n*n!;
```

$dp[m]$为所求解

5 数据结构

5.1 *Splay

持续学习中。
注意节点的size值不一定是真实的值！如果有需要需要特别维护！

1. 旋转和Splay操作
2. rank操作
3. insert操作（。。很多题目都有）
4. del操作（郁闷的出纳员）
5. 由数组建立Splay
6. 前驱后继（营业额统计）
7. Pushdown Pushup的位置
8. *。。。暂时想不起了

节点定义。。

```
const int MaxN = 50003;

struct Node
{
    int size,key;

    Node *c[2];
    Node *p;
} mem[MaxN], *cur, *nil;
```

无内存池的几个初始化函数。

```
Node *newNode(int v, Node *p)
{
    cur->c[0] = cur->c[1] = nil, cur->p = p;
    cur->size = 1;
    cur->key = v;
    return cur++;
}

void Init()
{
    cur = mem;
    nil = newNode(0, cur);
    nil->size = 0;
}
```

带内存池的几个函数。

```
int emp[MaxN], totemp;

Node *newNode(int v, Node *p)
{
    cur = mem + emp[--totemp];
    cur->c[0] = cur->c[1] = nil, cur->p = p;
    cur->size = 1;
    cur->key = v;
    return cur;
}

void Init()
{
    for (int i = 0; i < MaxN; ++i)
        emp[i] = i;
    totemp = MaxN;
    cur = mem + emp[--totemp];
    nil = newNode(0, cur);
    nil->size = 0;
}

void Recycle(Node *p)
{
    if (p == nil)   return;
    Recycle(p->c[0]), Recycle(p->c[1]);
    emp[totemp++] = p - mem;
}
```

基本的Splay框架。维护序列用。
一切下标从0开始。

```
struct SplayTree
{
    Node *root;
    void Init()
    {
        root = nil;
    }
    void Pushup(Node *x)
    {
        if (x == nil)   return;
        Pushdown(x); Pushdown(x->c[0]); Pushdown(x->c[1]);
        x->size = x->c[0]->size + x->c[1]->size + 1;
    }
    void Pushdown(Node *x)
    {
        if (x == nil)   return;
        //do something
    }
    void Rotate(Node *x, int f)
    {
        if (x == nil)   return;
        Node *y = x->p;
        y->c[f ^ 1] = x->c[f], x->p = y->p;
        if (x->c[f] != nil)
            x->c[f]->p = y;
        if (y->p != nil)
            y->p->c[y->p->c[1] == y] = x;
        x->c[f] = y, y->p = x;
        Pushup(y);
    }
    void Splay(Node *x, Node *f)
    {
        while (x->p != f)
        {
            Node *y = x->p;
            if (y->p == f)
                Rotate(x, x == y->c[0]);
            else
            {
                int fd = y->p->c[0] == y;
                if (y->c[fd] == x)
                    Rotate(x, fd ^ 1), Rotate(x, fd);
                else
                    Rotate(y, fd), Rotate(x, fd);
            }
        }
        Pushup(x);
        if (f == nil)
            root = x;
    }
    void Select(int k, Node *f)
    {
        Node *x = root;
        Pushdown(x);
        int tmp;
        while ((tmp = x->c[0]->size) != k)
        {
            if (k < tmp)    x = x->c[0];
            else
                x = x->c[1], k -= tmp + 1;
            Pushdown(x);
        }
        Splay(x, f);
    }
    void Select(int l, int r)
    {
        Select(l, nil), Select(r + 2, root);
    }
    Node *Make_tree(int a[], int l, int r, Node *p)
    {
        if (l > r)  return nil;
        int mid = l + r >> 1;
        Node *x = newNode(a[mid], p);
        x->c[0] = Make_tree(a, l, mid - 1, x);
        x->c[1] = Make_tree(a, mid + 1, r, x);
        Pushup(x);
        return x;
    }
    void Insert(int pos, int a[], int n)
    {
        Select(pos, nil), Select(pos + 1, root);
        root->c[1]->c[0] = Make_tree(a, 0, n - 1, root->c[1]);
        Splay(root->c[1]->c[0], nil);
    }
    void Insert(int v)
    {
        Node *x = root, *y = nil;
        while (x != nil)
        {
            y = x;
            y->size++;
            x = x->c[v >= x->key];
        }
        y->c[v >= y->key] = x = newNode(v, y);
        Splay(x, nil);
    }
    void Remove(int l, int r)
    {
        Select(l, r);
        //Recycle(root->c[1]->c[0]);
        root->c[1]->c[0] = nil;
        Splay(root->c[1], nil);
    }
};
```

例题：旋转区间赋值求和求最大子序列。
注意打上懒标记后立即Pushup。Pushup(root-c[1]-c[0]),Pushup(root-c[1]),Pushup(root);

```
void Pushup(Node *x)
{
  if (x == nil) return;
  Pushdown(x); Pushdown(x->c[0]); Pushdown(x->c[1]);
  x->size = x->c[0]->size+x->c[1]->size+1;

  x->sum = x->c[0]->sum+x->c[1]->sum+x->key;
  x->lsum = max(x->c[0]->lsum,x->c[0]->sum+x->key+max(0,x->c
     [1]->lsum));
  x->rsum = max(x->c[1]->rsum,x->c[1]->sum+x->key+max(0,x->c
     [0]->rsum));
  x->maxsum = max(max(x->c[0]->maxsum,x->c[1]->maxsum),x->key+
     max(0,x->c[0]->rsum)+max(0,x->c[1]->lsum));
}
void Pushdown(Node *x)
{
  if (x == nil) return;
```

```
    if (x->rev)
    {
      x->rev = 0;
      x->c[0]->rev ^= 1;
      x->c[1]->rev ^= 1;
      swap(x->c[0],x->c[1]);

      swap(x->lsum,x->rsum);
    }
    if (x->same)
    {
      x->same = false;
      x->key = x->lazy;
      x->sum = x->key*x->size;
      x->lsum = x->rsum = x->maxsum = max(x->key,x->sum);
      x->c[0]->same = true, x->c[0]->lazy = x->key;
      x->c[1]->same = true, x->c[1]->lazy = x->key;
    }
  }

int main()
{
  int totcas;
  scanf("%d",&totcas);
  for (int cas = 1;cas <= totcas;cas++)
  {
    Init();
    sp.Init();
    nil->lsum = nil->rsum = nil->maxsum = -Inf;
    sp.Insert(0);
    sp.Insert(0);

    int n,m;
    scanf("%d%d",&n,&m);
    for (int i = 0;i < n;i++)
      scanf("%d",&a[i]);
    sp.Insert(0,a,n);

    for (int i = 0;i < m;i++)
    {
      int pos,tot,c;
      scanf("%s",buf);
      if (strcmp(buf,"MAKE-SAME") == 0)
      {
        scanf("%d%d%d",&pos,&tot,&c);
        sp.Select(pos-1,pos+tot-2);
        sp.root->c[1]->c[0]->same = true;
        sp.root->c[1]->c[0]->lazy = c;
        sp.Pushup(sp.root->c[1]), sp.Pushup(sp.root);
      }
      else if (strcmp(buf,"INSERT") == 0)
      {
        scanf("%d%d",&pos,&tot);
        for (int i = 0;i < tot;i++)
          scanf("%d",&a[i]);
        sp.Insert(pos,a,tot);
      }
      else if (strcmp(buf,"DELETE") == 0)
      {
        scanf("%d%d",&pos,&tot);
        sp.Remove(pos-1,pos+tot-2);
      }
      else if (strcmp(buf,"REVERSE") == 0)
      {
        scanf("%d%d",&pos,&tot);
        sp.Select(pos-1,pos+tot-2);
        sp.root->c[1]->c[0]->rev ^= 1;
        sp.Pushup(sp.root->c[1]), sp.Pushup(sp.root);
      }
      else if (strcmp(buf,"GET-SUM") == 0)
      {
        scanf("%d%d",&pos,&tot);
        sp.Select(pos-1,pos+tot-2);
        printf("%d\n",sp.root->c[1]->c[0]->sum);
      }
      else if (strcmp(buf,"MAX-SUM") == 0)
      {
        sp.Select(0,sp.root->size-3);
        printf("%d\n",sp.root->c[1]->c[0]->maxsum);
      }
    }
  }
  return 0;
}
```

维护多个序列的时候，不需要建立很多Splay。只需要记录某个点在内存池中的绝对位置就可以了。
需要操作它所在的序列时直接Splay到nil。此时Splay的root所在的Splay就是这个序列了。
新建序列的时候需要多加入两个额外节点。如果某个Splay只有两个节点了需要及时回收。
例题：Box（维护括号序列）

```
    \\下面都是专用函数
    \\判断x在不在f里面
    bool Ancestor(Node *x,Node *f)
    {
        if (x == f) return true;
        while (x->p != nil)
        {
            if (x->p == f)  return true;
            x = x->p;
        }
        return false;
    }
    \\把Splay v插入到pos后面，pos=nil时新开一个序列
    void Insert(Node *pos, Node *v)
    {
        int pl;
        if (pos == nil)
        {
            Init();
            Insert(0), Insert(0);
            pl = 0;
        }
        else
        {
            Splay(pos, nil);
            pl = root->c[0]->size;
        }
        Select(pl, nil), Select(pl + 1, root);
        root->c[1]->c[0] = v;
        v->p = root->c[1];
        Splay(v, nil);
    }
    \\把[l,r]转出来（这里记录的是绝对位置）
    void Select(Node *l, Node *r)
    {
    Splay(l, nil);
        int pl = root->c[0]->size - 1;
        Splay(r, nil);
        int pr = root->c[0]->size - 1;
        Select(pl, pr);
    }
    \\分离[l,r]
    Node *Split(Node *l, Node *r)
    {
        Select(l, r);
        Node *res = root->c[1]->c[0];
        root->c[1]->c[0] = res->p = nil;
        Splay(root->c[1], nil);
        if (root->size == 2)
        {
            Recycle(root);
            Init();
        }
        return res;
    }

int main(int argc, char const *argv[])
{
    freopen("P.in", "r", stdin);
    bool first = true;
    while (scanf("%d", &n) != EOF)
    {
        if (!first) puts("");
        first = false;
        Init();
        for (int i = 0; i < n; i++)
        {
            \\建立独立的N个区间，记录绝对位置
            sp.Init();
            sp.Insert(0), sp.Insert(0);
            sp.Insert(0,i+1),sp.Insert(1,i+1);
            sp.Select(0, 0), l[i] = sp.root->c[1]->c[0];
            sp.Select(1, 1), r[i] = sp.root->c[1]->c[0];
        }
        for (int i = 0; i < n; i++)
        {
            int f;
            scanf("%d", &f);
            if (f != 0)
            {
                \\把[l[i],r[i]]插入到l[f-1]后面
                Node *pos = sp.Split(l[i], r[i]);
                sp.Insert(l[f - 1], pos);
            }
        }
        scanf("%d", &n);
        for (int i = 0; i < n; i++)
        {
            scanf("%s", com);
            if (com[0] == 'Q')
            {
                int pos;
                scanf("%d", &pos);
                \\求[l[pos-1],r[pos-1]]在哪个序列里面
                sp.Splay(l[pos - 1], nil);
                sp.Select(1, nil);
                printf("%d\n", sp.root->key);
            }
            else
            {
                int u, v;
                scanf("%d%d", &u, &v);
                if (v == 0)
                    sp.Insert(nil, sp.Split(l[u-1], r[u-1]));
                else
                {
                    sp.Select(l[u-1],r[u-1]);
                    if (sp.Ancestor(l[v-1],sp.root->c[1]->c[0]) == false)
                        sp.Insert(l[v - 1], sp.Split(l[u-1], r[u-1]));
                }
            }
        }
    }
    return 0;
}
```

5.2 动态树

懒标记是否及时Pushdown了？
修改之后有没有及时Pushup？

5.2.1 维护点权

查询链上的最长字段和
GetRoute是用换根写的

```
const int MaxN = 110000;

struct Node
{
    int size, key;
    bool rev;

//     bool same;
//     int lsum, rsum, sum, maxsum, sa;

    Node *c[2];
    Node *p;
} mem[MaxN], *cur, *nil, *pos[MaxN];

Node *newNode(int v, Node *p)
{
    cur->c[0] = cur->c[1] = nil, cur->p = p;
    cur->size = 1;
    cur->key = v;
    cur->rev = false;

//     cur->same = false;
//     cur->sa = 0;
//     cur->lsum = cur->rsum = cur->maxsum = 0;
//     cur->sum = v;

    return cur++;
}

void Init()
{
    cur = mem;
    nil = newNode(0, cur);
    nil->size = 0;
}

struct SplayTree
{
    void Pushup(Node *x)
    {
        if (x == nil)   return;
        Pushdown(x); Pushdown(x->c[0]); Pushdown(x->c[1]);
        x->size = x->c[0]->size + x->c[1]->size + 1;

//          x->sum = x->c[0]->sum + x->c[1]->sum + x->key;
//          x->lsum = max(x->c[0]->lsum, x->c[0]->sum + x->key +
    max(0, x->c[1]->lsum));
//          x->rsum = max(x->c[1]->rsum, x->c[1]->sum + x->key +
    max(0, x->c[0]->rsum));
//          x->maxsum = max(max(x->c[0]->maxsum, x->c[1]->maxsum),
//              x->key + max(0, x->c[0]->rsum) + max(0, x->c[1]->
    lsum));

    }
    void Pushdown(Node *x)
    {
        if (x == nil)   return;
        if (x->rev)
        {
            x->rev = 0;
            x->c[0]->rev ^= 1;
            x->c[1]->rev ^= 1;
            swap(x->c[0], x->c[1]);
//注意修改与位置有关的量
//              swap(x->lsum,x->rsum);
        }

//          if (x->same)
//          {
//              x->same = false;
//              x->key = x->sa;
//              x->sum = x->sa * x->size;
//              x->lsum = x->rsum = x->maxsum = max(0, x->sum);
//              if (x->c[0] != nil)
//                  x->c[0]->same = true, x->c[0]->sa = x->sa;
//              if (x->c[1] != nil)
//                  x->c[1]->same = true, x->c[1]->sa = x->sa;
//          }
    }
    bool isRoot(Node *x)
    {
        return (x == nil) || (x->p->c[0] != x && x->p->c[1] != x)
            ;
    }
    void Rotate(Node *x, int f)
    {
        if (isRoot(x))   return;
        Node *y = x->p;
        y->c[f ^ 1] = x->c[f], x->p = y->p;
        if (x->c[f] != nil)
            x->c[f]->p = y;
        if (y != nil)
        {
            if (y == y->p->c[1])
```

```
                y->p->c[1] = x;
            else if (y == y->p->c[0])
                y->p->c[0] = x;
        }
        x->c[f] = y, y->p = x;
        Pushup(y);
    }
    void Splay(Node *x)
    {
        static Node *stack[MaxN];
        int top = 0;
        stack[top++] = x;
        for (Node *y = x; !isRoot(y); y = y->p)
            stack[top++] = y->p;
        while (top)
            Pushdown(stack[--top]);

        while (!isRoot(x))
        {
            Node *y = x->p;
            if (isRoot(y))
                Rotate(x, x == y->c[0]);
            else
            {
                int fd = y->p->c[0] == y;
                if (y->c[fd] == x)
                    Rotate(x, fd ^ 1), Rotate(x, fd);
                else
                    Rotate(y, fd), Rotate(x, fd);
            }
        }
        Pushup(x);
    }
    Node *Access(Node *u)
    {
        Node *v = nil;
        while (u != nil)
        {
            Splay(u);
            v->p = u;
            u->c[1] = v;
            Pushup(u);
            u = (v = u)->p;
            if (u == nil)
                return v;
        }
    }
    Node *LCA(Node *u, Node *v)
    {
        Access(u);
        return Access(v);
    }
    Node *Link(Node *u, Node *v)
    {
        Access(u);
        Splay(u);
        u->rev = true;
        u->p = v;
    }
    void ChangeRoot(Node *u)
    {
        Access(u)->rev ^= 1;
    }
    Node *GetRoute(Node *u, Node *v)
    {
        ChangeRoot(u);
        return Access(v);
    }
};

int n, m;
SplayTree sp;

int main(int argc, char const *argv[])
{
    while (scanf("%d", &n) != EOF)
    {
        Init();
        for (int i = 0; i < n; i++)
        {
            int v;
            scanf("%d", &v);
            pos[i] = newNode(v, nil);
        }
        for (int i = 0; i < n - 1; i++)
        {
            int u, v;
            scanf("%d%d", &u, &v);
            u--, v--;
            sp.Link(pos[u], pos[v]);
        }

//          scanf("%d", &m);
//          for (int i = 0; i < m; i++)
//          {
//              int typ, u, v, c;
//              scanf("%d%d%d", &typ, &u, &v);
//              u--, v--;
//              if (typ == 1)
//                  printf("%d\n", sp.GetRoute(pos[u], pos[v])->
    maxsum);
//              else
//              {
//                  scanf("%d", &c);
//                  Node *p = sp.GetRoute(pos[u], pos[v]);
//                  p->same = true;
//                  p->sa = c;
//              }
//          }
    }
    return 0;
}
```

5.2.2 维护边权

刘汝佳的Happy Painting!
查询链上边的不同颜色数量
不能换根，但是可以Link和Cut

```
const int MaxN = 60000;

struct Node
{
    int size,key;

    int msk,lazy;

    Node *c[2];
    Node *p;
} mem[MaxN], *cur, *nil, *pos[MaxN];

Node *newNode(int v,Node *p)
{
    cur->c[0] = cur->c[1] = nil, cur->p = p;
    cur->size = 1;
    cur->key = v;

    cur->msk = 0;
    cur->lazy = -1;

    return cur++;
}

void Init()
{
    cur = mem;
    nil = newNode(0, cur);
    nil->size = 0;
}

struct SplayTree
{
    void Pushup(Node *x)
    {
        if (x == nil) return;
        Pushdown(x);
        Pushdown(x->c[0]);
        Pushdown(x->c[1]);
        x->size = x->c[0]->size + x->c[1]->size + 1;

        x->msk = x->c[0]->msk | x->c[1]->msk | (1<<x->key);
    }
    void Pushdown(Node *x)
    {
        if (x == nil) return;

        if (x->lazy != -1)
        {
            x->key = x->lazy;
            x->msk = (1<<x->key);
            x->c[0]->lazy = x->c[1]->lazy = x->lazy;
            x->lazy = -1;
        }
    }
    bool isRoot(Node *x)
    {
        return (x == nil) || (x->p->c[0] != x && x->p->c[1] != x)
          ;
    }
    void Rotate(Node *x, int f)
    {
        if (isRoot(x))  return;
        Node *y = x->p;
        y->c[f ^ 1] = x->c[f], x->p = y->p;
        if (x->c[f] != nil)
            x->c[f]->p = y;
        if (y != nil)
        {
            if (y == y->p->c[1])
                y->p->c[1] = x;
            else if (y == y->p->c[0])
                y->p->c[0] = x;
        }
        x->c[f] = y, y->p = x;
        Pushup(y);
    }
    void Splay(Node *x)
    {
        static Node *stack[MaxN];
        int top = 0;
        stack[top++] = x;
        for (Node *y = x; !isRoot(y); y = y->p)
            stack[top++] = y->p;
        while (top)
            Pushdown(stack[--top]);

        while (!isRoot(x))
        {
            Node *y = x->p;
            if (isRoot(y))
                Rotate(x, x == y->c[0]);
            else
            {
                int fd = y->p->c[0] == y;
                if (y->c[fd] == x)
                    Rotate(x, fd ^ 1), Rotate(x, fd);
                else
                    Rotate(y, fd), Rotate(x, fd);
            }
        }
        Pushup(x);
    }
    Node *Access(Node *u)
    {
        Node *v = nil;
        while (u != nil)
        {
            Splay(u);
            v->p = u;
            u->c[1] = v;
            Pushup(u);
            u = (v = u)->p;
            if (u == nil) return v;
        }
    }
    Node *Root(Node *u)
    {
        Access(u);
        Splay(u);
        for (Pushdown(u); u->c[0] != nil; u = u->c[0])
            Pushdown(u);
        Splay(u);
        return u;
    }
    Node *LCA(Node *u, Node *v)
    {
        if (Root(u) != Root(v))
            return nil;
        Access(u);
        return Access(v);
    }
    void Cut(Node *u)
    {
        Access(u);
        Splay(u);
        u->c[0] = u->c[0]->p = nil;
        Pushup(u);
    }
    void Link(Node *u, Node *v, int val)
    {
        Access(u);
        Splay(u);
        u->p = v;
        u->key = val;
        Pushup(u);
    }
};

int cntbit(int x)
{
    x = (x & 0x55555555) + ((x >> 1) & 0x55555555);
    x = (x & 0x33333333) + ((x >> 2) & 0x33333333);
    x = (x & 0x0F0F0F0F) + ((x >> 4) & 0x0F0F0F0F);
    x = (x & 0x00FF00FF) + ((x >> 8) & 0x00FF00FF);
    x = (x & 0x0000FFFF) + ((x >> 16) & 0x0000FFFF);
    return x;
}

SplayTree sp;
int n,Q,f[MaxN];

int main(int argc, char const *argv[])
{
    while (scanf("%d%d",&n,&Q) != EOF)
    {
        Init();
        for (int i = 0; i < n; i++)
        {
            scanf("%d",&f[i]);
            pos[i] = newNode(0, nil);
        }
        for (int i = 0; i < n; i++)
        {
            int col;
            scanf("%d",&col);
            if (f[i] > 0)
                sp.Link(pos[i],pos[f[i]-1],col-1);
        }
        for (int q = 0; q < Q; q++)
        {
            int typ,x,y,c;
            scanf("%d%d%d",&typ,&x,&y);
            x--,y--;
            if (typ == 3)
            {
                Node *lca = sp.LCA(pos[x],pos[y]);
                if (lca == nil || x == y)
                {
                    printf("0␣0\n");
                    continue;
                }
                int totedge = lca->c[1]->size;
                int msk = lca->c[1]->msk;

                if (pos[x] != lca)
                {
                    sp.Splay(pos[x]);
                    totedge += pos[x]->size;
                    msk |= pos[x]->msk;
                }

                printf("%d␣%d\n",totedge,cntbit(msk));
            }
            else
            {
                scanf("%d",&c);
                c--;
                if (typ == 1)
                {
                    if (x == y) continue;

                    Node *lca = sp.LCA(pos[x],pos[y]);
                    if (pos[x] == lca)  continue;

                    sp.Cut(pos[x]);
                    sp.Link(pos[x],pos[y],c);

                }
                else
                {
                    Node *lca = sp.LCA(pos[x],pos[y]);

                    if (lca == nil || x == y)
                        continue;

                    lca->c[1]->lazy = c;
                    sp.Pushup(lca->c[1]);
                    sp.Pushup(lca);
                    if (pos[x] != lca)
                    {
                        sp.Splay(pos[x]);
                        pos[x]->lazy = c;
                        sp.Pushup(pos[x]);
                    }
                }
            }
        }
    }
    return 0;
}
```

5.3 可持久化线段树

区间第k小数，内存压缩版，POJ2014。

```
#include <cstdio>
#include <algorithm>
using namespace std;

const int MAXN=100000,MAXM=100000;

struct node
{
    node *l,*r;
    int sum;
}tree[MAXN*4+MAXM*20];

int N;
node *newnode()
{
    tree[N].l=tree[N].r=NULL;
    tree[N].sum=0;
    return &tree[N++];
}
node *newnode(node *x)
{
    tree[N].l=x->l;
    tree[N].r=x->r;
    tree[N].sum=x->sum;
    return &tree[N++];
}
node *build(int l,int r)
{
    node *x=newnode();
    if (l<r)
    {
        int mid=l+r>>1;
        x->l=build(l,mid);
        x->r=build(mid+1,r);
        x->sum=x->l->sum+x->r->sum;
    }
    else
        x->sum=0;
    return x;
}
node *update(node *x,int l,int r,int p,int v)
{
    if (l<r)
    {
        int mid=l+r>>1;
        node *nx=newnode(x);
        if (p<=mid)
        {
            node *ret=update(x->l,l,mid,p,v);
            nx->l=ret;
        }
        else
        {
            node *ret=update(x->r,mid+1,r,p,v);
            nx->r=ret;
        }
        nx->sum=nx->l->sum+nx->r->sum;
        return nx;
    }
    else
    {
        node *nx=newnode(x);
        nx->sum+=v;
        return nx;
    }
}
int query(node *x1,node *x2,int l,int r,int k)
{
    if (l<r)
    {
        int mid=l+r>>1;
        int lsum=x2->l->sum-x1->l->sum;
        if (lsum>=k)
            return query(x1->l,x2->l,l,mid,k);
        else
            return query(x1->r,x2->r,mid+1,r,k-lsum);
    }
    else
        return l;
}
char s[10];
node *root[MAXM+1];
int a[MAXN],b[MAXN];
int init(int n)
{
    for (int i=0;i<n;i++)
        b[i]=a[i];
    sort(b,b+n);
    int tn=unique(b,b+n)-b;
    for (int i=0;i<n;i++)
    {
        int l=0,r=tn-1;
        while (l<r)
        {
            int mid=l+r>>1;
            if (b[mid]>=a[i])
                r=mid;
            else
                l=mid+1;
        }
        a[i]=l;
    }
    return tn;
}
```

```
int main()
{
    int cas=1,n;
    while (scanf("%d",&n)!=EOF)
    {
        printf("Case␣%d:\n",cas++);
        for (int i=0;i<n;i++)
            scanf("%d",&a[i]);
        int tn=init(n);
        N=0;
        root[0]=build(0,tn-1);
        for (int i=1;i<=n;i++)
            root[i]=update(root[i-1],0,tn-1,a[i-1],1);
        int m;
        scanf("%d",&m);
        for (int i=0;i<m;i++)
        {
            int s,t;
            scanf("%d%d",&s,&t);
            printf("%d\n",b[query(root[s-1],root[t],0,tn-1,t-s
               +2>>1)]);
        }
    }
    return 0;
}
```

5.4 treap正式版

支持翻转。

```
#include <cstdio>
#include <cstdlib>
#include <algorithm>
using namespace std;

const int MAXN = 100000;
const int MAXM = 100000;
const int inf = 0x7fffffff;
int a[MAXN];
struct Treap
{
    int N;
    Treap()
    {
        N = 0;
        root = NULL;
    }
    void init()
    {
        N = 0;
        root = NULL;
    }
    struct Treap_Node
    {
        Treap_Node *son[2];//left & right
        int value, fix;
        bool lazy;
        int size;
        Treap_Node() {}
        Treap_Node(int _value)
        {
            son[0] = son[1] = NULL;
            value = _value;
            fix = rand() * rand();
            lazy = 0;
            size = 1;
        }
        int sonSize(bool flag)
        {
            if (son[flag] == NULL)
                return 0;
            else
                return son[flag]->size;
        }
    } node[MAXN], *root, *pos[MAXN];
    void up(Treap_Node *p)
    {
        p->size = p->sonSize(0) + p->sonSize(1) + 1;
    }
    void down(Treap_Node *p)
    {
        if (!p->lazy)
            return ;
        for (int i = 0; i < 2; i++)
            if (p->son[i])
                p->son[i]->lazy = !p->son[i]->lazy;
        swap(p->son[0], p->son[1]);
        p->lazy = 0;
    }
    Treap_Node *merge(Treap_Node *p, Treap_Node *q)
    {
        if (p == NULL)
            return q;
        else if (q == NULL)
            return p;
        if (p->fix <= q->fix)
        {
            down(p);
            p->son[1] = merge(p->son[1], q);
            up(p);
            return p;
        }
        else
        {
            down(q);
            q->son[0] = merge(p, q->son[0]);
            up(q);
            return q;
        }
    }
    pair<Treap_Node *, Treap_Node *> split(Treap_Node *p, int n)
    {
        if (p == NULL)
            return make_pair((Treap_Node *)NULL, (Treap_Node *)NULL);
        if (!n)
            return make_pair((Treap_Node *)NULL, p);
        if (n == p->size)
            return make_pair(p, (Treap_Node *)NULL);
        down(p);
        if (p->sonSize(0) >= n)
        {
            pair<Treap_Node *, Treap_Node *> ret = split(p->son[0], n);
            p->son[0] = ret.second;
            up(p);
            return make_pair(ret.first, p);
        }
        else
        {
            pair<Treap_Node *, Treap_Node *> ret = split(p->son[1], n - p->sonSize(0) - 1);
            p->son[1] = ret.first;
            up(p);
            return make_pair(p, ret.second);
        }
    }
    int smalls(Treap_Node *p,int value)
    {
        if (p==NULL)
            return 0;
        if (p->value<=value)
            return 1+p->sonSize(0)+smalls(p->son[1],value);
        else
            return smalls(p->son[0],value);
    }
    void insert(int value)
    {
        Treap_Node *p = &node[N++];
        *p = Treap_Node(value);
        pair<Treap_Node *, Treap_Node *> ret = split(root, smalls(root, value));
        root = merge(merge(ret.first, p), ret.second);
    }
    void remove(int value)
    {
        pair<Treap_Node *, Treap_Node *> ret = split(root, smalls
           (root, value) - 1);
        root = merge(ret.first, split(ret.second, 1).second);
    }
    Treap_Node *build(int s, int t)
    {
        int idx = t + s >> 1;
        Treap_Node *p = &node[N++];
        *p = Treap_Node(a[idx]);
        pos[a[idx]] = p;
        if (idx > s)
            p = merge(build(s, idx - 1), p);
        if (idx < t)
            p = merge(p, build(idx + 1, t));
        up(p);
        return p;
    }
    void build(int n)
    {
        root = build(0, n - 1);
    }
    void *reverse(int s, int t)
    {
        pair<Treap_Node *, Treap_Node *> tmp1, tmp2;
        tmp1 = split(root, s - 1);
        tmp2 = split(tmp1.second, t - s + 1);
        tmp2.first->lazy = !tmp2.first->lazy;
        root = merge(tmp1.first, merge(tmp2.first, tmp2.second));
    }
};
Treap treap;
int main()
{
    treap.init();
    int n;
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
        scanf("%d", &a[i]);
    treap.build(n);
}
```

5.5 树链剖分

5.5.1 点权

```
#include <cstdio>
#include <cstring>
#include <cstdlib>
#include <algorithm>
using namespace std;
const int MAX = 12000;
const int LOG = 15;
const int oo = 0x3f3f3f3f;
struct Edge
{
        int to, w, id;
        Edge* next;
} memo[MAX << 1], *cur, *g[MAX], *pree[MAX], *solid[MAX], *valid[
   MAX];
int dp[MAX][LOG], pos[MAX], lst[MAX], dep[MAX], cnt[MAX], h[MAX],
    K, n;
void init()
{
    for (int i = 1; i <= n; i++)
    {
        g[i] = NULL;
        valid[i] = NULL;
        solid[i] = NULL;
        pree[i] = NULL;
    }
    for (int i = 0; i < LOG; i++)
    {
        dp[1][i] = 1;
    }
    cur = memo;
    K = 0;
}
void add(int u, int v, int w, int id)
{
    cur->to = v;
    cur->w = w;
    cur->id = id;
    cur->next = g[u];
    g[u] = cur++;
}
void dfsLCA(int d, int u, int f)
{
    dep[u] = d;
    dp[u][0] = f;
    cnt[u] = 1;
    for (int i = 1; i < LOG; i++)
    {
        dp[u][i] = dp[dp[u][i - 1]][i - 1];
    }
    for (Edge* it = g[u]; it; it = it->next)
    {
        int v = it->to;
        if (v != f)
        {
            pree[v] = it;
            valid[it->id] = it;
            dfsLCA(d + 1, v, u); //RE
            cnt[u] += cnt[v];
            if (solid[u] == NULL || cnt[solid[u]->to] < cnt[v])
            {
                solid[u] = it;
            }
        }
    }
}
void dfsChain(int u, int head)
{
    h[u] = head;
    if (solid[u])
    {
        lst[pos[u] = K++] = u;
        dfsChain(solid[u]->to, head);
    }
    else
    for (Edge* it = g[u]; it; it = it->next)
    {
        int v = it->to;
        if (it != solid[u] && v != dp[u][0])
        {
            dfsChain(v, v);
        }
    }
}
int getLCA(int u, int v)
{
    if (dep[u] < dep[v])
        swap(u, v);
    for (int st = 1 << (LOG - 1), i = LOG - 1; i >= 0; i--, st
       >>= 1)
    {
        if (st <= dep[u] - dep[v])
        {
            u = dp[u][i];
        }
    }
    if (u == v)
        return u;
    for (int i = LOG - 1; i >= 0; i--)
    {
        if (dp[u][i] != dp[v][i])
        {
            u = dp[u][i];
            v = dp[v][i];
        }
    }
    return dp[u][0];
}
struct Node
{
        int l, r, ma, mi;
        bool rev;
```

```
} seg[MAX << 2];
void reverse(int k)
{
    seg[k].mi *= -1;
    seg[k].ma *= -1;
    seg[k].rev ^= 1;
    swap(seg[k].mi, seg[k].ma);
}
void pushdown(int k)
{
    if (seg[k].rev)
    {
        reverse(k << 1);
        reverse(k << 1 | 1);
        seg[k].rev = false;
    }
}
void update(int k)
{
    seg[k].mi = min(seg[k << 1].mi, seg[k << 1 | 1].mi);
    seg[k].ma = max(seg[k << 1].ma, seg[k << 1 | 1].ma);
}
void init(int k, int l, int r)
{
    seg[k].l = l;
    seg[k].r = r;
    seg[k].rev = false;
    if (l == r)
    {
        seg[k].mi = seg[k].ma = solid[lst[l]]->w; //solid WA
        return;
    }
    int mid = l + r >> 1;
    init(k << 1, l, mid);
    init(k << 1 | 1, mid + 1, r);
    update(k);
}
void update(int k, int id, int v)
{
    if (seg[k].l == seg[k].r)
    {
        seg[k].mi = seg[k].ma = solid[lst[id]]->w = v;
        return;
    }
    pushdown(k);
    int mid = seg[k].l + seg[k].r >> 1;
    if (id <= mid)
        update(k << 1, id, v);
    else
        update(k << 1 | 1, id, v);
    update(k);
}
```

```
void reverse(int k, int l, int r)
{
    if (seg[k].l > r || seg[k].r < l)
        return;
    if (seg[k].l >= l && seg[k].r <= r)
    {
        reverse(k);
        return;
    }
    pushdown(k);
    reverse(k << 1, l, r);
    reverse(k << 1 | 1, l, r);
    update(k);
}
int read(int k, int l, int r)
{
    if (seg[k].l > r || seg[k].r < l)
        return -oo;
    if (seg[k].l >= l && seg[k].r <= r)
        return seg[k].ma;
    pushdown(k);
    return max(read(k << 1, l, r), read(k << 1 | 1, l, r));
}
void setEdge(int id, int v)
{
    Edge* it = valid[id];
    if (h[it->to] != it->to)
    {
        update(1, pos[dp[it->to][0]], v);
    }
    else
    {
        it->w = v;
    }
}
void negateLCA(int t, int u)
{
    while (t != u)
    {
        int tmp = h[u];
        if (dep[tmp] < dep[t])
            tmp = t;
        if (h[u] == u)
        {
            pree[u]->w *= -1;
            u = dp[u][0];
        }
        else
        {
            reverse(1, pos[tmp], pos[dp[u][0]]);
            u = tmp;
        }
    }
}
void negate(int u, int v)
{
    int t = getLCA(u, v);
    negateLCA(t, u);
    negateLCA(t, v);
}
int maxLCA(int t, int u)
{
    int ret = -oo;
    while (t != u)
    {
        int tmp = h[u];
        if (dep[tmp] < dep[t])
            tmp = t;
        if (h[u] == u)
        {
            ret = max(ret, pree[u]->w);
            u = dp[u][0];
        }
        else
        {
            ret = max(ret, read(1, pos[tmp], pos[dp[u][0]]));
            u = tmp;
        }
    }
    return ret;
}
int query(int u, int v)
{
    int t = getLCA(u, v);
    return max(maxLCA(t, u), maxLCA(t, v));
}
int main()
{
    int T;
    int u, v, w;
    char op[15];
    scanf("%d", &T);
    while (T--)
    {
        scanf("%d", &n);
        init();
        for (int i = 1; i < n; i++)
        {
            scanf("%d%d%d", &u, &v, &w);
            add(u, v, w, i);
            add(v, u, w, i);
        }
        dfsLCA(0, 1, 1);
        dfsChain(1, 1);
        init(1, 0, K - 1);
        while (scanf("%s", op), op[0] != 'D')
        {
            scanf("%d%d", &u, &v);
            if (op[0] == 'C')
            {
                setEdge(u, v);
            }
            else if (op[0] == 'N')
            {
                negate(u, v);
            }
            else
            {
                printf("%d\n", query(u, v));
            }
        }
    }
    return 0;
}
```

5.5.2 边权

```
#include <cstdio>
#include <iostream>
#include <cstdlib>
#include <algorithm>
#include <cmath>
#include <cstring>
using namespace std;
int n,m,sum,pos;
int head[50005],e;
int s[50005],from[50005];
int fa[50005][20],deep[50005],num[50005];
int solid[50005],p[50005],fp[50005];
struct N
{
  int l,r,mid;
  int add,w;
}nod[50005*4];
struct M
{
  int v,next;
}edge[100005];
void addedge(int u,int v)
{
  edge[e].v=v;
  edge[e].next=head[u];
  head[u]=e++;

  edge[e].v=u;
  edge[e].next=head[v];
  head[v]=e++;
```

```
}
void LCA(int st,int f,int d)
{
  deep[st]=d;
  fa[st][0]=f;
  num[st]=1;
  int i,v;
  for(i=1;i<20;i++)
    fa[st][i]=fa[fa[st][i-1]][i-1];
  for(i=head[st];i!=-1;i=edge[i].next)
  {
    v=edge[i].v;
    if(v!=f)
    {
      LCA(v,st,d+1);
      num[st]+=num[v];
      if(solid[st]==-1||num[v]>num[solid[st]])
        solid[st]=v;
    }
  }
}
void getpos(int st,int sp)
{
  from[st]=sp;
  if(solid[st]!=-1)
  {
    p[st]=pos++;
    fp[p[st]]=st;
    getpos(solid[st],sp);
  }
  else
  {
    p[st]=pos++;
    fp[p[st]]=st;
    return;
  }
  int i,v;
  for(i=head[st];i!=-1;i=edge[i].next)
  {
    v=edge[i].v;
    if(v!=solid[st]&&v!=fa[st][0])
      getpos(v,v);
  }
}
int getLCA(int u,int v)
{
  if(deep[u]<deep[v])
    swap(u,v);
  int d=1<<19,i;
  for(i=19;i>=0;i--)
  {
    if(d<=deep[u]-deep[v])
```

```
      u=fa[u][i];
    d>>=1;
  }
  if(u==v)
    return u;
  for(i=19;i>=0;i--)
    if(fa[u][i]!=fa[v][i])
    {
      u=fa[u][i];
      v=fa[v][i];
    }
  return fa[u][0];
}
void init(int p,int l,int r)
{
  nod[p].l=l;
  nod[p].r=r;
  nod[p].mid=(l+r)>>1;
  nod[p].add=0;
  if(l==r)
    nod[p].w=s[fp[l]];
  else
  {
    init(p<<1,l,nod[p].mid);
    init(p<<1|1,nod[p].mid+1,r);
  }
}
void lazy(int p)
{
  if(nod[p].add!=0)
  {
    nod[p<<1].add+=nod[p].add;
    nod[p<<1|1].add+=nod[p].add;
    nod[p].add=0;
  }
}
void update(int p,int l,int r,int v)
{
  if(nod[p].l==l&&nod[p].r==r)
  {
    nod[p].add+=v;
    return;
  }
  lazy(p);
  if(nod[p].mid<l)
    update(p<<1|1,l,r,v);
  else if(nod[p].mid>=r)
    update(p<<1,l,r,v);
  else
  {
    update(p<<1,l,nod[p].mid,v);
    update(p<<1|1,nod[p].mid+1,r,v);
  }
}
int read(int p,int l,int r)
{
  if(nod[p].l==l&&nod[p].r==r)
    return nod[p].w+nod[p].add;
  lazy(p);
  if(nod[p].mid<l)
    return read(p<<1|1,l,r);
  else if(nod[p].mid>=r)
    return read(p<<1,l,r);
}
void jump(int st,int ed,int val)
{
  while(deep[st]>=deep[ed])
  {
    int tmp=from[st];
    if(deep[tmp]<deep[ed])
      tmp=ed;
    update(1,p[tmp],p[st],val);
    st=fa[tmp][0];
  }
}
void change(int st,int ed,int val)
{
  int lca=getLCA(st,ed);
  jump(st,lca,val);
  jump(ed,lca,val);
  jump(lca,lca,-val);
}
int main()
{
  while(scanf("%d%d%d",&n,&m,&sum)==3)
  {
    int i;
    s[0]=0;pos=0;deep[0]=-1;
    memset(fa,0,sizeof(fa));
    for(i=1;i<=n;i++)
    {
      solid[i]=-1;
      scanf("%d",&s[i]);
    }
    memset(head,-1,sizeof(head));
    e=0;
    for(i=0;i<m;i++)
    {
      int a,b;
      scanf("%d%d",&a,&b);
      addedge(a,b);
    }
    LCA(1,0,0);
    getpos(1,1);
    init(1,0,pos-1);
    for(i=0;i<sum;i++)
    {
      char que[5];
      scanf("%s",que);
      if(que[0]!='Q')
      {
        int a,b,c;
        scanf("%d%d%d",&a,&b,&c);
        if(que[0]=='D')
          c=-c;
        change(a,b,c);
      }
      else
      {
        int a;
        scanf("%d",&a);
        printf("%d\n",read(1,p[a],p[a]));
      }
    }
  }
  return 0;
}
```

5.6 划分树

```
int n,m;
struct elem
{
    int v,index;
}a[120000];
int d[30][120000];
int s[30][120000];

bool cmp(elem a,elem b)
{
    if (a.v == b.v)
        return a.index <= b.index;
    return a.v < b.v;
}

void build(int depth,int l,int r)
{
    if (l == r)
        return;
    int mid = (l+r)/2;
    int tl,tr;
    tl = tr = 0;
    for (int i = l;i <= r;i++)
    {
        if (cmp(a[d[depth][i]],a[mid]))
        {
            d[depth+1][l+tl] = d[depth][i];
```

```
                tl++;
            }
            else
            {
                d[depth+1][mid+1+tr] = d[depth][i];
                tr++;
            }
            s[depth][i] = tl;
        }
        build(depth+1,l,mid);
        build(depth+1,mid+1,r);
}

int find(int depth,int dl,int dr,int fl,int fr,int k)
{
        if (fl == fr)
            return a[d[depth][fl]].v;
        int ls,rs;
        int mid = (dl+dr)/2;
        ls = (fl == dl)? 0 : s[depth][fl-1];
        rs = s[depth][fr];
        return (rs-ls < k)? find(depth+1,mid+1,dr,mid+fl-dl-ls+1,mid+
           fr-dl-rs+1,k-(rs-ls)) : find(depth+1,dl,mid,dl+ls,dl+rs-1,
           k);
}

int main()
{
        while (scanf("%d%d",&n,&m) != EOF)
        {
            for (int i = 1;i <= n;i++)
            {
                scanf("%d",&a[i].v);
                a[i].index = i;
            }
            sort(a+1,a+n+1,cmp);
            for (int i = 1;i <= n;i++)
                d[0][a[i].index] = i;
            build(0,1,n);
            int l,r,k;
            for (int i = 1;i <= m;i++)
            {
                scanf("%d%d%d",&l,&r,&k);
                printf("%d\n",find(0,1,n,l,r,k));
            }
        }
        return 0;
}
```

5.7 树状数组

```
int read(int k)
{
    int sum = 0;
    for (; k; k^=k&-k)
        sum+=tree[k];
    return sum;
}
void update(int k, int v)
{
    for (; k<=MaxN; k+=k&-k)
        tree[k]+=v;
}
int find_Kth(int k)
{
    int idx = 0;
    for(int i=20; i>=0; i--)
    {
        idx |= 1 << i;
        if(idx <= MaxN && tree[idx] < k)
            k -= tree[idx];
        else  idx ^= 1 << i;
    }
    return idx + 1;
}
```

6 图论

6.1 优先队列优化的dijkstra

```
#include<cstdio>
#include<cstring>
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
using namespace std;
const int MAXN=100;
const int MAXM=1000;
int N,L;
int head[MAXN];
struct edges
{
    int to,next,cost;
} edge[MAXM];
int dist[MAXN];
class states
{
public:
    int cost,id;
};
class cmp
{
public:
    bool operator ()(const states &i,const states &j)
    {
        return i.cost>j.cost;
    }
};
void init(int n)
{
    N=n;
    L=0;
    for (int i=0; i<n; i++)
        head[i]=-1;
}
void add_edge(int x,int y,int cost)
{
    edge[L].to=y;
    edge[L].cost=cost;
    edge[L].next=head[x];
    head[x]=L++;
}
int dijkstra(int s,int t)
{
    memset(dist,63,sizeof(dist));
    states u;
    u.id=s;
    u.cost=0;
    dist[s]=0;
    priority_queue<states,vector<states>,cmp> q;
    q.push(u);
    while (!q.empty())
    {
        u=q.top();
        q.pop();
        if (u.id==t) return dist[t];
        if (u.cost!=dist[u.id]) continue;
        for (int i=head[u.id]; i!=-1; i=edge[i].next)
        {
            states v=u;
            v.id=edge[i].to;
            if (dist[v.id]>dist[u.id]+edge[i].cost)
            {
                v.cost=dist[v.id]=dist[u.id]+edge[i].cost;
                q.push(v);
            }
        }
    }
    return -1;
}
int main()
{
    int n,m;
    scanf("%d%d",&n,&m);
    init(n);
    for (int i=0; i<m; i++)
    {
        int x,y,z;
        scanf("%d%d%d",&x,&y,&z);
        add_edge(x,y,z);
        add_edge(y,x,z);
    }
    int s,t;
    scanf("%d%d",&s,&t);
    printf("%d\n",dijkstra(s,t));
    return 0;
}
```

6.2 SAP四版

```
const int MAXEDGE=20400;
const int MAXN=400;
const int inf=0x3fffffff;
struct edges
{
    int cap,to,next,flow;
} edge[MAXEDGE+100];
struct nodes
{
    int head,label,pre,cur;
```

```
} node[MAXN+100];
int L,N;
int gap[MAXN+100];
void init(int n)
{
    L=0;
    N=n;
    for (int i=0; i<N; i++)
        node[i].head=-1;
}
void add_edge(int x,int y,int z,int w)
{
    edge[L].cap=z;
    edge[L].flow=0;
    edge[L].to=y;
    edge[L].next=node[x].head;
    node[x].head=L++;
    edge[L].cap=w;
    edge[L].flow=0;
    edge[L].to=x;
    edge[L].next=node[y].head;
    node[y].head=L++;
}
int maxflow(int s,int t)
{
    memset(gap,0,sizeof(gap));
    gap[0]=N;
    int u,ans=0;
    for (int i=0; i<N; i++)
    {
        node[i].cur=node[i].head;
        node[i].label=0;
    }
    u=s;
    node[u].pre=-1;
    while (node[s].label<N)
    {
        if (u==t)
        {
            int min=inf;
            for (int i=node[u].pre; i!=-1; i=node[edge[i^1].to].
               pre)
              if (min>edge[i].cap-edge[i].flow)
                  min=edge[i].cap-edge[i].flow;
            for (int i=node[u].pre; i!=-1; i=node[edge[i^1].to].
               pre)
            {
              edge[i].flow+=min;
              edge[i^1].flow-=min;
            }
            u=s;
            ans+=min;
```

```
                continue;
            }
            bool flag=false;
            int v;
            for (int i=node[u].cur; i!=-1; i=edge[i].next)
            {
                v=edge[i].to;
                if (edge[i].cap-edge[i].flow && node[v].label+1==node
                    [u].label)
                {
                    flag=true;
                    node[u].cur=node[v].pre=i;
                    break;
                }
            }
            if (flag)
            {
                u=v;
                continue;
            }
            node[u].cur=node[u].head;
            int min=N;
            for (int i=node[u].head; i!=-1; i=edge[i].next)
                if (edge[i].cap-edge[i].flow && node[edge[i].to].
                    label<min)
                    min=node[edge[i].to].label;
            gap[node[u].label]--;
            if (!gap[node[u].label]) return ans;
            node[u].label=min+1;
            gap[node[u].label]++;
            if (u!=s) u=edge[node[u].pre^1].to;
        }
        return ans;
}
```

6.3 费用流三版

T了可以改成栈。

```
const int MAXM=60000;
const int MAXN=400;
const int inf=0x3fffffff;
int L,N;
int K;
struct edges
{
    int to,next,cap,flow,cost;
} edge[MAXM];
struct nodes
{
    int dis,pre,head;
    bool visit;
} node[MAXN];
```

```
void init(int n)
{
    N=n;
    L=0;
    for (int i=0; i<N; i++)
        node[i].head=-1;
}
void add_edge(int x,int y,int cap,int cost)
{
    edge[L].to=y;
    edge[L].cap=cap;
    edge[L].cost=cost;
    edge[L].flow=0;
    edge[L].next=node[x].head;
    node[x].head=L++;
    edge[L].to=x;
    edge[L].cap=0;
    edge[L].cost=-cost;
    edge[L].flow=0;
    edge[L].next=node[y].head;
    node[y].head=L++;
}
bool spfa(int s,int t)
{
    queue <int> q;
    for (int i=0; i<N; i++)
    {
        node[i].dis=0x3fffffff;
        node[i].pre=-1;
        node[i].visit=0;
    }
    node[s].dis=0;
    node[s].visit=1;
    q.push(s);
    while (!q.empty())
    {
        int u=q.front();
        node[u].visit=0;
        for (int i=node[u].head; i!=-1; i=edge[i].next)
        {
            int v=edge[i].to;
            if (edge[i].cap>edge[i].flow &&
                    node[v].dis>node[u].dis+edge[i].cost)
            {
                node[v].dis=node[u].dis+edge[i].cost;
                node[v].pre=i;
                if (!node[v].visit)
                {
                    node[v].visit=1;
                    q.push(v);
                }
            }
        }
        q.pop();
    }
    if (node[t].pre==-1)
        return 0;
    else
        return 1;
}
int mcmf(int s,int t,int &cost)
{
    int flow=0;
    while (spfa(s,t))
    {
        int max=inf;
        for (int i=node[t].pre; i!=-1; i=node[edge[i^1].to].pre)
        {
            if (max>edge[i].cap-edge[i].flow)
                max=edge[i].cap-edge[i].flow;
        }
        for (int i=node[t].pre; i!=-1; i=node[edge[i^1].to].pre)
        {
            edge[i].flow+=max;
            edge[i^1].flow-=max;
            cost+=edge[i].cost*max;
        }
        flow+=max;
    }
    return flow;
}
```

6.4 匈牙利

6.4.1 新版,隐式图可解

```
bool check(int u)
{
    for (int i=head[u]; i!=-1; i=edge[i].next)
    {
        int v=edge[i].to;
        if (matc[v]==u) continue;
        if (!use[v])
        {
            use[v]=1;
            if (matc[v]==-1 || check(matc[v]))
            {
                matc[v]=u;
                matc[u]=v;
                return 1;
            }
        }
    }
    return 0;
}
int match()
{
    int ret=0;
    memset(matc,-1,sizeof(matc));
    for (int u=0; u<N; u++)
    {
        if (matc[u]!=-1) continue;
        memset(use,0,sizeof(use));
        if (check(u))
            ret++;
    }
    return ret;
}
```

6.4.2 邻接矩阵

```
bool check(int u)
{
    for (int v=0; v<N; v++)
        if (am[u][v] && !use[v])
        {
            use[v]=1;
            if (pre[v]==-1 || check(pre[v]))
            {
                pre[v]=u;
                return 1;
            }
        }
    return 0;
}
int match()
{
    int ret=0;
    memset(pre,-1,sizeof(pre));
    for (int u=0; u<N; u++)
    {
        memset(use,0,sizeof(use));
        if (check(u))
            ret++;
    }
    return ret;
}
```

6.4.3 邻接表

```
bool check(int u)
{
    for (int i=head[u]; i!=-1; i=edge[i].next)
    {
        int v=edge[i].to;
        if (!use[v])
        {
            use[v]=1;
            if (pre[v]==-1 || check(pre[v]))
            {
                pre[v]=u;
                return 1;
            }
        }
    }
    return 0;
}
int match()
{
    int ret=0;
    memset(pre,-1,sizeof(pre));
    for (int u=1; u<=N; u++)
    {
        memset(use,0,sizeof(use));
        if (check(u))
            ret++;
    }
    return ret;
}
```

6.5 一般图匹配带花树

```
const int MaxN = 222;
int N;
bool Graph[MaxN+1][MaxN+1];
int Match[MaxN+1];
bool InQueue[MaxN+1],InPath[MaxN+1],InBlossom[MaxN+1];
int Head,Tail;
int Queue[MaxN+1];
int Start,Finish;
int NewBase;
int Father[MaxN+1],Base[MaxN+1];
int Count;
void CreateGraph()
{
    int u,v;
    memset(Graph,false,sizeof(Graph));
    scanf("%d",&N);
    while (scanf("%d%d",&u,&v) != EOF)
        Graph[u][v] = Graph[v][u] = true;
}
void Push(int u)
{
    Queue[Tail] = u;
    Tail++;
    InQueue[u] = true;
}
int Pop()
{
    int res = Queue[Head];
    Head++;
    return res;
}
int FindCommonAncestor(int u,int v)
{
    memset(InPath,false,sizeof(InPath));
    while (true)
    {
        u = Base[u];
        InPath[u] = true;
        if (u == Start) break;
        u = Father[Match[u]];
    }
    while (true)
    {
        v = Base[v];
        if (InPath[v]) break;
        v = Father[Match[v]];
    }
    return v;
}
void ResetTrace(int u)
{
    int v;
    while (Base[u] != NewBase)
    {
        v = Match[u];
        InBlossom[Base[u]] = InBlossom[Base[v]] = true;
        u = Father[v];
        if (Base[u] != NewBase) Father[u] = v;
    }
}
void BlossomContract(int u,int v)
{
    NewBase = FindCommonAncestor(u,v);
    memset(InBlossom,false,sizeof(InBlossom));
    ResetTrace(u);
    ResetTrace(v);
    if (Base[u] != NewBase) Father[u] = v;
    if (Base[v] != NewBase) Father[v] = u;
    for (int tu = 1; tu <= N; tu++)
        if (InBlossom[Base[tu]])
        {
            Base[tu] = NewBase;
            if (!InQueue[tu]) Push(tu);
        }
}
void FindAugmentingPath()
{
    memset(InQueue,false,sizeof(InQueue));
    memset(Father,0,sizeof(Father));
    for (int i = 1; i <= N; i++)
        Base[i] = i;
    Head = Tail = 1;
    Push(Start);
    Finish = 0;
    while (Head < Tail)
    {
        int u = Pop();
        for (int v = 1; v <= N; v++)
            if (Graph[u][v] && (Base[u] != Base[v]) && (Match[u]
               != v))
            {
                if ((v == Start) || ((Match[v] > 0) && (Father[
                   Match[v]] > 0)))
                    BlossomContract(u,v);
                else if (Father[v] == 0)
                {
                    Father[v] = u;
                    if (Match[v] > 0)
                        Push(Match[v]);
                    else
                    {
                        Finish = v;
                        return;
                    }
                }
            }
    }
}
void AugmentPath()
{
    int u,v,w;
    u = Finish;
    while (u > 0)
    {
        v = Father[u];
        w = Match[v];
        Match[v] = u;
        Match[u] = v;
        u = w;
    }
}
void Edmonds()
{
    memset(Match,0,sizeof(Match));
    for (int u = 1; u <= N; u++)
        if (Match[u] == 0)
        {
            Start = u;
            FindAugmentingPath();
            if (Finish > 0) AugmentPath();
        }
}
void PrintMatch()
```

```
{
    for (int u = 1; u <= N; u++)
        if (Match[u] > 0)
            Count++;
    printf("%d\n",Count);
    for (int u = 1; u <= N; u++)
        if (u < Match[u])
            printf("%d␣%d\n",u,Match[u]);
}
int main()
{
    CreateGraph();
    Edmonds();
    PrintMatch();
}
```

6.6 KM

6.6.1 最大加权匹配

```
bool visx[N],visy[N];//x,y中的点是否被访问
int lx[N],ly[N];//x,y中的点的标号
int matchy[N];//y中各点匹配状态
int map[N][N];//二分图描述[x][y]
bool find(int x)
{
  visx[x]=true;
  int t;
  for (int y=0;y<ycnt;y++)
  {
    if (!visy[y])
    {
      t=lx[x]+ly[y]-map[x][y];
      if (t==0)
      {
        visy[y]=true;
        if (matchy[y]==-1 || find(matchy[y]))
        {
          matchy[y]=x;
          return true;
        }
      }
      else if (lack>t) lack=t;
    }
  }
  return false;
}
void KM()
{
  memset(lx,0,sizeof(lx));
  memset(ly,0,sizeof(ly));
  memset(matchy,-1,sizeof(matchy));
  for (int i=0;i<xcnt;i++)
```

```
    for (int j=0;j<ycnt;j++)
      if (map[i][j]>lx[i])
        lx[i]=map[i][j];
  for (int x=0;x<xcnt;x++)
  {
    while (true)
    {
      memset(visx,false,sizeof(visx));
      memset(visy,false,sizeof(visy));
      lack=INFI;
      if (find(x)) break;
      for (int i=0;i<xcnt;i++)
      {
        if (visx[i]) lx[i]-=lack;
        if (visy[i]) ly[i]+=lack;
      }
    }
  }
  int cost=0;
  for (int i=0;i<ycnt;i++)
    cost+=map[matchy[i]][i];
}
```

6.6.2 自认为正确的Kuhn_Munkras

未验证

```
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
const int MAXN=100;
const int inf=0x3f3f3f3f;
bool visitx[MAXN],visity[MAXN];
int labx[MAXN],laby[MAXN],matx[MAXN],maty[MAXN],slack[MAXN];
int ma[MAXN][MAXN];
bool check(int x,int n)
{
    visitx[x]=1;
    for (int i=0; i<n; i++)
        if (!visity[i])
            if (labx[x]+laby[i]==ma[x][i])
            {
                visity[i]=1;
                if (maty[i]==-1 || check(maty[i],n))
                {
                    matx[x]=i;
                    maty[i]=x;
                    return 1;
                }
            }
            else
                slack[i]=min(slack[i],labx[x]+laby[i]-ma[x][i]);

    return 0;
}
void maintain(int n)
{
    int diff=inf;
    for (int i=0; i<n; i++)
        if (!visity[i])
            diff=min(diff,slack[i]);
    for (int i=0; i<n; i++)
    {
        if (visitx[i])
            labx[i]-=diff;
        if (visity[i])
            laby[i]+=diff;
        else
            slack[i]-=diff;
    }
}
int Kuhn_Munkras(int n)
{
    for (int i=0; i<n; i++)
    {
        labx[i]=-inf;
        for (int j=0; j<n; j++)
            labx[i]=max(labx[i],ma[i][j]);
    }
    memset(laby,0,4*n);
    memset(matx,-1,4*n);
    memset(maty,-1,4*n);
    for (int i=0; i<n; i++)
    {
        memset(visitx,0,n);
        memset(visity,0,n);
        memset(slack,63,4*n);
        while (!check(i,n))
        {
            maintain(n);
            memset(visitx,0,n);
            memset(visity,0,n);
        }
    }
    int ret=0;
    for (int i=0;i<n;i++)
        ret+=labx[i]+laby[i];
    return ret;
}
int main()
{
    int n,m;
    scanf("%d%d",&m,&n);
    for (int i=m; i<n; i++)
        for (int j=0; j<n; j++)
```

```
                ma[i][j]=0;
        for (int i=0; i<m; i++)
            for (int j=0; j<n; j++)
                scanf("%d",&ma[i][j]);
        printf("%d\n",Kuhn_Munkras(n));
        printf("%d",matx[0]+1);
        for (int i=1;i<m;i++)
            printf("␣%d",matx[i]+1);
        puts("");
        return 0;
}
```

6.7 *二维平面图的最大流

待整理

```
#include <iostream>
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <vector>
#include <cmath>
#include <map>
#include <queue>
using namespace std;

const int maxn = 100100;
const int inf = 0x3f3f3f3f;
struct Point
{
    int x,y,id;
    double theta;
    Point() {}
    Point(int _x,int _y)
    {
        x = _x;
        y = _y;
    }
    Point(Point _s,Point _e,int _id)
    {
        id = _id;
        x = _s.x-_e.x;
        y = _s.y-_e.y;
        theta = atan2(y,x);
    }
    bool operator < (const Point &b)const
    {
        return theta < b.theta;
    }
};

map<pair<int,int>,int > idmap;
```

```
struct Edge
{
    int from,to,next,cap,near,mark;
};
Edge edge[maxn*2];
int head[maxn],L;
int cntd[maxn];
void addedge(int u,int v,int cap)
{
    cntd[u]++;
    cntd[v]++;
    idmap[make_pair(u,v)] = L;
    edge[L].from = u;
    edge[L].to = v;
    edge[L].cap = cap;
    edge[L].next = head[u];
    edge[L].mark = -1;
    head[u] = L++;
}

int rtp[maxn];
Point p[maxn],tp[maxn];
int n,m,S,T;
int vid;

struct Edge2
{
    int to,next,dis;
} edge2[maxn*2];
int head2[maxn],L2;

void addedge2(int u,int v,int dis)
{
    edge2[L2].to = v;
    edge2[L2].dis = dis;
    edge2[L2].next = head2[u];
    head2[u] = L2++;
}

int dist[maxn];
bool inq[maxn];
int SPFA(int s,int t)
{
    queue<int> Q;
    memset(inq,false,sizeof(inq));
    memset(dist,63,sizeof(dist));
    Q.push(s);
    dist[s] = 0;
    while (!Q.empty())
    {
        int now = Q.front();
        Q.pop();
        for (int i = head2[now]; i != -1; i = edge2[i].next)
            if (dist[edge2[i].to] > dist[now]+edge2[i].dis)
            {
                dist[edge2[i].to] = dist[now]+edge2[i].dis;
                if (inq[edge2[i].to] == false)
                {
                    inq[edge2[i].to] = true;
                    Q.push(edge2[i].to);
                }
            }
        inq[now] = false;
    }
    return dist[t];
}

int main()
{
    int totcas;
    scanf("%d",&totcas);
    for (int cas = 1; cas <= totcas; cas++)
    {
        idmap.clear();
        L = 0;
        scanf("%d%d",&n,&m);
        S = T = 0;
        for (int i = 0; i < n; i++)
        {
            head[i] = -1;
            scanf("%d%d",&p[i].x,&p[i].y);
            if (p[S].x > p[i].x)
                S = i;
            if (p[T].x < p[i].x)
                T = i;
            cntd[i] = 0;
        }
        //源汇中间加入一个特殊节点
        head[n] = -1;
        n ++;
        addedge(S,n-1,inf);
        addedge(n-1,S,inf);
        addedge(T,n-1,inf);
        addedge(n-1,T,inf);

        for (int i = 0; i < m; i++)
        {
            int u,v,cap;
            scanf("%d%d%d",&u,&v,&cap);
            u--;
            v--;
            addedge(u,v,cap);
            addedge(v,u,cap);
        }

        for (int i = 0; i < n; i++)
        {
            int tot = 0;
            //源点汇点连到特殊点的方向需要特别考虑一下
            if (i == S)
                tp[tot++] = Point(Point(0,0),Point(-1,0),n-1);
            else if (i == T)
                tp[tot++] = Point(Point(0,0),Point(1,0),n-1);
            else if (i == n-1)
            {
                tp[tot++] = Point(Point(0,0),Point(1,0),S);
                tp[tot++] = Point(Point(0,0),Point(-1,0),T);
            }
            if (i < n-1)
            {
                for (int j = head[i]; j != -1; j = edge[j].next)
                {
                    if (i == S && edge[j].to == n-1)  continue;
                    if (i == T && edge[j].to == n-1)  continue;
                    tp[tot++] = Point(p[i],p[edge[j].to],edge[j].
                       to);
                }
            }
            sort(tp,tp+tot);
            for (int j = 0; j < tot; j++)
                rtp[tp[j].id] = j;
            for (int j = head[i]; j != -1; j = edge[j].next)
                edge[j].near = tp[(rtp[edge[j].to]+1)%tot].id;
        }

        vid = 0;
        for (int i = 0;i < L;i++)
            if (edge[i].mark == -1)
            {
                int now = edge[i].from;
                int eid = i;
                int to  = edge[i].to;
                while (true)
                {
                    edge[eid].mark = vid;
                    eid ^= 1;
                    now = to;
                    to = edge[eid].near;
                    eid = idmap[make_pair(now,to)];

                    if (now == edge[i].from)    break;
                }
                vid++;
            }

        L2 = 0;
        for (int i = 0; i < vid; i++)
            head2[i] = -1;
        for (int i = 0; i < L; i++)
            addedge2(edge[i].mark,edge[i^1].mark,edge[i].cap);
        printf("%d\n",SPFA(edge[0].mark,edge[1].mark));
    }
    return 0;
}
```

6.8 强联通

```
int dfsnum[2000];
int low[2000];
int stack[2000];
int top;
int ans;
int an;
int be[2000];
int flag[2000];
void dfs(int x)
{
    dfsnum[x] = low[x] = ans++;
    stack[++top] = x;
    flag[x] = 1;
    for (int i = head[x];i != -1;i = edge[i].next)
    {
        int y = edge[i].to;
        if (dfsnum[y] == -1)
        {
            dfs(y);
            low[x] = min(low[x],low[y]);
        }
        else if (flag[y] == 1)
            low[x] = min(low[x],dfsnum[y]);
    }
    if (dfsnum[x] == low[x])
    {
        while (stack[top] != x)
        {
            flag[stack[top]] = 0;
            be[stack[top]] = an;
            top--;
        }
        flag[x] = 0;
        be[x] = an++;
        top--;
    }
}
```

调用:

```
void SC()
```

```
{
    memset(dfsnum,-1,sizeof(dfsnum));
    memset(flag,0,sizeof(flag));
    top = 0;
    an = 0;
    ans = 0;
    for (int i = 0;i < n;i++)
        if (dfsnum[i] == -1)
            dfs(i);
}
```

6.9 最大团以及相关知识

独立集: 独立集是指图的顶点集的一个子集,该子集的导出子图不含边.如果一个独立集不是任何一个独立集的子集, 那么称这个独立集是一个极大独立集.一个图中包含顶点数目最多的独立集称为最大独立集。最大独立集 一定是极大独立集，但是极大独立集不一定是最大的独立集。

支配集: 与独立集相对应的就是支配集，支配集也是图顶点集的一个子集，设S是图G的一个支配集，则对于图中的任意一个顶点u，要么属于集合s, 要么与s中的顶点相邻。在s中除去任何元素后s不再是支配集，则支配集s是极小支配集。称G的所有支配集中顶点个数最 少的支配集为最小支配集，最小支配集中的顶点个数成为支配数。

最小点的覆盖: 最小点的覆盖也是图的顶点集的一个子集，如果我们选中一个点，则称这个点将以他为端点的所有边都覆盖了。将图中所有的边都覆盖所用顶点数最少，这个集合就是最小的点的覆盖。

最大团: 图G的顶点的子集，设D是最大团，则D中任意两点相邻。若u，v是最大团，则u,v有边相连，其补图u,v没有边相连，所以图G的最大团=其补图的最大独立集。给定无向图$G = (V, E)$，如果U属于V，并且对于任意u,v包含于U 有$< u, v >$包含于E，则称U是G的完全子图，G的完全子图U是G的团，当且仅当U不包含在G的更大的完全子图中，G的最大团是指G中所含顶点数目最多的团。如果U属于V，并且对于任意u, v包含于U有$< u, v >$不包含于E，则称U是G的空子图，G的空子图U是G的独立集，当且仅当U不包含在G的更大的独立集，G的最大团是指G中所含顶点数目最多的独立集。

一些性质: 最大独立集+最小覆盖集$= V$，最大团=补图的最大独立集，最小覆盖集=最大匹配

```
#include <cstdio>
bool am[100][100];
int ans;
int c[100];
int U[100][100];
int n;
bool dfs(int rest,int num)
{
    if (!rest)
    {
        if (num>=ans)
            return 1;
        else
            return 0;
    }
    int pre=-1;
    for (int i=0;i<rest && rest-i+num>=ans;i++)
    {
        int idx=U[num][i];
        if (num+c[idx]<ans)
            return 0;
        int nrest=0;
        for (int j=i+1; j<rest; j++)
            if (am[idx][U[num][j]])
                U[num+1][nrest++]=U[num][j];
        if (dfs(nrest,num+1))
            return 1;
    }
    return 0;
}
int main()
{
    while (scanf("%d",&n),n)
    {
        for (int i=0;i<n;i++)
            for (int j=0;j<n;j++)
                scanf("%d",&am[i][j]);
        ans=0;
        for (int i=n-1; i>=0; i--)
        {
            int rest=0;
            for (int j=i+1; j<n; j++)
                if (am[i][j])
                    U[0][rest++]=j;
            ans+=dfs(rest,0);
            c[i]=ans;
        }
        printf("%d\n",ans);
    }
    return 0;
}
```

6.10 双连通分量

标号从0起

```
#include<cstdio>
#include<cstring>
#include<stack>
#include<queue>
#include<algorithm>
using namespace std;
const int MAXN=100000*2;
const int MAXM=200000;
struct edges
{
    int to,next;
    bool cut,visit;
```

```
} edge[MAXM<<1];
int head[MAXN],low[MAXN],dpt[MAXN],L;
bool visit[MAXN],cut[MAXN];
void init(int n)
{
    L=0;
    memset(head,-1,4*n);
    memset(visit,0,n);
}
void add_edge(int u,int v)
{
    edge[L].cut=edge[L].visit=0;
    edge[L].to=v;
    edge[L].next=head[u];
    head[u]=L++;
}
int idx;
stack<int> st;
int bcc[MAXM];
void dfs(int u,int fu,int deg)
{
    cut[u]=0;
    visit[u]=1;
    low[u]=dpt[u]=deg;
    int tot=0;
    for (int i=head[u]; i!=-1; i=edge[i].next)
    {
        int v=edge[i].to;
        if (edge[i].visit)
            continue;
        st.push(i/2);
        edge[i].visit=edge[i^1].visit=1;
        if (visit[v])
        {
            low[u]=dpt[v]>low[u]?low[u]:dpt[v];
            continue;
        }
        dfs(v,u,deg+1);
        edge[i].cut=edge[i^1].cut=(low[v]>dpt[u] || edge[i].cut);
        if (u!=fu) cut[u]=low[v]>=dpt[u]?1:cut[u];
        if (low[v]>=dpt[u] || u==fu)
        {
            while (st.top()!=i/2)
            {
                int x=st.top()*2,y=st.top()*2+1;
                bcc[st.top()]=idx;
                st.pop();
            }
            bcc[i/2]=idx++;
            st.pop();
        }
        low[u]=low[v]>low[u]?low[u]:low[v];
        tot++;
    }
    if (u==fu && tot>1) cut[u]=1;
}
int main()
{
    int n,m;
    while (scanf("%d%d",&n,&m)!=EOF)
    {
        init(n);
        for (int i=0; i<m; i++)
        {
            int u,v;
            scanf("%d%d",&u,&v);
            add_edge(u,v);
            add_edge(v,u);
        }
        idx=0;
        for (int i=0; i<n; i++)
            if (!visit[i])
                dfs(i,i,0);
    }
    return 0;
}
```

6.11 割点与桥

```
#include<cstdio>
#include<cstring>
const int MAXN=10000;
struct edges
{
    int to,next;
    bool cut,visit;
    int from;
} edge[MAXN-1<<1];
int head[MAXN],low[MAXN],dfn[MAXN],L;
bool visit[MAXN],cut[MAXN];
void init(int n)
{
    L=0;
    memset(head,-1,4*n);
    memset(cut,0,4*n);
    memset(visit,0,4*n);
}
void add_edge(int u,int v)
{
    edge[L].from=u;
    edge[L].cut=edge[L].visit=0;
    edge[L].to=v;
    edge[L].next=head[u];
    head[u]=L++;
}
int idx;
void dfs(int u,int fu)
{
    visit[u]=1;
    low[u]=dfn[u]=idx++;
    int tot=0;
    for (int i=head[u]; i!=-1; i=edge[i].next)
    {
        int v=edge[i].to;
        if (edge[i].visit)
            continue;
        edge[i].visit=edge[i^1].visit=1;
        if (visit[v])
        {
            low[u]=dfn[v]>low[u]?low[u]:dfn[v];
            continue;
        }
        dfs(v,u);
        edge[i].cut=edge[i^1].cut=low[v]>dfn[u] || edge[i].cut;
        if (u!=fu) cut[u]=low[v]>=dfn[u]?1:cut[u];
        low[u]=low[v]>low[u]?low[u]:low[v];
        tot++;
    }
    if (u==fu && tot>1) cut[u]=1;
}
int main()
{
    int t;
    scanf("%d",&t);
    while (t--)
    {
        int n,m;
        scanf("%d%d",&n,&m);
        init(n);
        for (int i=0; i<m; i++)
        {
            int u,v;
            scanf("%d%d",&u,&v);
            add_edge(--u,--v);
            add_edge(v,u);
        }
        for (int i=0; i<n; i++)
            if (!visit[i])
            {
                idx=0;
                dfs(i,i);
            }
    }
    return 0;
}
```

6.12 LCA

在线LCA，bfs

```
#include<cstdio>
#include<cstring>
#include<queue>
using namespace std;
const int NSIZE = 50000;
const int DEG = 20;
struct trees
{

    int fa[DEG];
    int head,deg;
} tree[NSIZE];
struct edges
{
    int to , next;
} edge[NSIZE];
struct states
{
    int u,fu,deg;
};
int L;
void add_edge(int x, int y)
{
    edge[L].to = y;
    edge[L].next = tree[x].head;
    tree[x].head = L++;
}
int Root;
void BFS(int s)
{
    queue<states> que;
    states st;
    st.deg=0;
    st.fu=st.u=s;
    que.push(st);
    while(!que.empty())
    {
        states st=que.front();
        que.pop();
        tree[st.u].deg = st.deg;
        tree[st.u].fa[0] = st.fu;
        for (int i=1;i<DEG;i++)
            tree[st.u].fa[i]=s;
        for (int tmp=st.fu,num=1;tree[tmp].deg;tmp=tree[st.u].fa[
           num++])
            tree[st.u].fa[num]=tree[tmp].fa[num-1];
        for(int i = tree[st.u].head ; i != -1; i = edge[i].next)
        {
            int v = edge[i].to;
            if (v == st.fu) continue;
            states nst;
            nst.u=v;
            nst.fu=st.u;
            nst.deg=st.deg+1;
            que.push(nst);
        }
    }
}
int LCA(int x, int y)
{
    if(tree[x].deg > tree[y].deg) swap(x,y);
    int hx=tree[x].deg,hy=tree[y].deg;
    int tx=x,ty=y;
    for (int det=hy-hx,i=0; det; det>>=1,i++)
        if (det&1)
            ty=tree[ty].fa[i];
    if(tx == ty) return tx;
    for (int i=DEG-1; i>=0; i--)
    {
        if(tree[tx].fa[i] == tree[ty].fa[i])
            continue;
        tx = tree[tx].fa[i];
        ty = tree[ty].fa[i];
    }
    return tree[tx].fa[0];
}
int main()
{
    int t;
    scanf("%d",&t);
    while(t--)
    {
        int n;
        scanf("%d",&n);
        L = 0;
        for(int i = 0 ; i < n ; i++)
            tree[i].head = -1;
        for(int i = 0 ; i < n-1 ; i++)
        {
            int a,b;
            scanf("%d%d",&a ,&b);
            add_edge(a-1,b-1);
            add_edge(b-1,a-1);
        }
        Root=0;
        BFS(Root);
        int a,b;
        scanf("%d%d",&a,&b);
        int lca=LCA(a-1,b-1)+1;
        printf("%d\n",lca);
    }
    return 0;
}
```

6.13 最优比例生成树

```
#include<stdio.h>
#include<string.h>
#include<math.h>
struct
{
    int x,y;
    double z;
} node[1100];
struct
{
    double l,c;
} map[1100][1100];
int n,l,f[1100],pre[1100];
double dis[1100];
double mst(double x)
{
    int i,j,tmp;
    double min,s=0,t=0;
    memset(f,0,sizeof(f));
    f[1]=1;
    for (i=2; i<=n; i++)
    {
        dis[i]=map[1][i].c-map[1][i].l*x;
        pre[i]=1;
    }
    for (i=1; i<n; i++)
    {
        min=1e10;
        for (j=1; j<=n; j++)
            if (!f[j] && min>dis[j])
            {
                min=dis[j];
                tmp=j;
            }
        f[tmp]=1;
        t+=map[pre[tmp]][tmp].l;
        s+=map[pre[tmp]][tmp].c;
        for (j=1; j<=n; j++)
            if (!f[j] && map[tmp][j].c-map[tmp][j].l*x<dis[j])
            {
                dis[j]=map[tmp][j].c-map[tmp][j].l*x;
                pre[j]=tmp;
            }
    }
    return s/t;
}
int main()
```

```
{
    int i,j;
    double a,b;
    scanf("%d",&n);
    while (n)
    {
        for (i=1; i<=n; i++)
            scanf("%d%d%lf",&node[i].x,&node[i].y,&node[i].z);
        for (i=1; i<=n; i++)
            for (j=i+1; j<=n; j++)
            {
                map[j][i].l=map[i][j].l=sqrt(1.0*(node[i].x-node[
                   j].x)*(node[i].x-node[j].x)+(node[i].y-node[j
                   ].y)*(node[i].y-node[j].y));
                map[j][i].c=map[i][j].c=fabs(node[i].z-node[j].z)
                   ;
            }
        a=0,b=mst(a);
        while (fabs(b-a)>1e-8)
        {
            a=b;
            b=mst(a);
        }
        printf("%.3f\n",b);
        scanf("%d",&n);
    }
}
```

6.14 全局最小割

```
#include <iostream>
using namespace std;
const int maxn=510;
int map[maxn][maxn];
int n;
void contract(int x,int y)
{
    int i,j;
    for (i=0; i<n; i++)
        if (i!=x) map[x][i]+=map[y][i],map[i][x]+=map[i][y];
    for (i=y+1; i<n; i++) for (j=0; j<n; j++)
        {
            map[i-1][j]=map[i][j];
            map[j][i-1]=map[j][i];
        }
    n--;
}
int w[maxn],c[maxn];
int sx,tx;
int mincut()
{
    int i,j,k,t;
    memset(c,0,sizeof(c));
    c[0]=1;
    for (i=0; i<n; i++) w[i]=map[0][i];
    for (i=1; i+1<n; i++)
    {
        t=k=-1;
        for (j=0; j<n; j++) if (c[j]==0&&w[j]>k)
                k=w[t=j];
        c[sx=t]=1;
        for (j=0; j<n; j++) w[j]+=map[t][j];
    }
    for (i=0; i<n; i++) if (c[i]==0) return w[tx=i];
}
int main()
{
    int i,j,k,m;
    while (scanf("%d%d",&n,&m)!=EOF)
    {
        memset(map,0,sizeof(map));
        while (m--)
        {
            scanf("%d%d%d",&i,&j,&k);
            map[i][j]+=k;
            map[j][i]+=k;
        }
        int mint=999999999;
        while (n>1)
        {
            k=mincut();
            if (k<mint) mint=k;
            contract(sx,tx);
        }
        printf("%d\n",mint);
    }
    return 0;
}
```

6.15 欧拉路

6.15.1 有向图

```
void solve(int x)
{
    int i;
    if (!match[x])
    {
        path[++l]=x;
        return ;
    }
    for (i=1; i<=n; i++)
        if (b[x][i])
        {
            b[x][i]--;
            match[x]--;
            solve(i);
        }
    path[++l]=x;
}
```

6.15.2 无向图

```
void solve(int x)
{
    int i;
    if (!match[x])
    {
        path[++l]=x;
        return ;
    }
    for (i=1; i<=n; i++)
        if (b[x][i])
        {
            b[x][i]--;
            b[i][x]--;
            match[x]--;
            match[i]--;
            solve(i);
        }
    path[++l]=x;
}
```

6.15.3 混合图

zju1992

```
int in[MAXN+100],out[MAXN+100];
int main()
{
    int t;
    scanf("%d",&t);
    while (t--)
    {
        int n,m;
        scanf("%d%d",&n,&m);
        N=n+2;L=-1;
        for (int i=0;i<N;i++)
            head[i]=-1;
        memset(in,0,sizeof(in));
        memset(out,0,sizeof(out));

        for (int i=0;i<m;i++)
        {
            int x,y,z;
            scanf("%d%d%d",&x,&y,&z);
            in[y]++;out[x]++;
            if (!z)
                add_edge(x,y,1);
        }
        int flag=1;
        for (int i=1;i<=n;i++)
        {
            if (in[i]-out[i]>0)
                add_edge(i,n+1,(in[i]-out[i])/2);
            else
            if (out[i]-in[i]>0)
                add_edge(0,i,(out[i]-in[i])/2);
            //printf("%d %d %d\n",i,out[i],in[i]);
            if ((in[i]+out[i])&1)
            {
                flag=0;
                break;
            }
        }
        maxflow(0,n+1);
        for (int i=head[0];i!=-1;i=edge[i].next)
            if (edge[i].cap>0 && edge[i].cap>edge[i].flow)
            {
                flag=0;
                break;
            }
        if (flag)
            puts("possible");
        else
            puts("impossible");
    }
    return 0;
}
```

6.16 K短路

```
#include<cstdio>
#include<cstring>
#include<queue>
using namespace std;
int K;
class states
{
public:
    int cost,id;
};
int dist[1000];
class cmp
{
public:
    bool operator ()(const states &i,const states &j)
    {
        return i.cost>j.cost;
    }
};
class cmp2
{
```

```
public:
    bool operator ()(const states &i,const states &j)
    {
        return i.cost+dist[i.id]>j.cost+dist[j.id];
    }
};
struct edges
{
    int to,next,cost;
} edger[100000],edge[100000];
int headr[1000],head[1000],Lr,L;
void dijkstra(int s)
{
    states u;
    u.id=s;
    u.cost=0;
    dist[s]=0;
    priority_queue<states,vector<states>,cmp> q;
    q.push(u);
    while (!q.empty())
    {
        u=q.top();
        q.pop();
        if (u.cost!=dist[u.id]) continue;
        for (int i=headr[u.id]; i!=-1; i=edger[i].next)
        {
            states v=u;
            v.id=edger[i].to;
            if (dist[v.id]>dist[u.id]+edger[i].cost)
            {
                v.cost=dist[v.id]=dist[u.id]+edger[i].cost;
                q.push(v);
            }
        }
    }
}
int num[1000];
void init(int n)
{
    Lr=L=0;
    memset(head,-1,4*n);
    memset(headr,-1,4*n);
    memset(dist,63,4*n);
    memset(num,0,4*n);
}
void add_edge(int u,int v,int x)
{
    edge[L].to=v;
    edge[L].cost=x;
    edge[L].next=head[u];
    head[u]=L++;
    edger[Lr].to=u;
    edger[Lr].cost=x;
    edger[Lr].next=headr[v];
    headr[v]=Lr++;
}
int a_star(int s,int t)
{
    if (dist[s]==0x3f3f3f3f)
        return -1;
    priority_queue<states,vector<states>,cmp2> q;
    states tmp;
    tmp.id=s;
    tmp.cost=0;
    q.push(tmp);
    while (!q.empty())
    {
        states u=q.top();
        q.pop();
        num[u.id]++;
        if (num[t]==K)
            return u.cost;
        for (int i=head[u.id]; i!=-1; i=edge[i].next)
        {
            int v=edge[i].to;
            tmp.id=v;
            tmp.cost=u.cost+edge[i].cost;
            q.push(tmp);
        }
    }
    return -1;
}
int main()
{
    int n,m;
    scanf("%d%d",&n,&m);
    init(n);
    for (int i=0; i<m; i++)
    {
        int u,v,x;
        scanf("%d%d%d",&u,&v,&x);
        add_edge(u-1,v-1,x);
    }
    int s,t;
    scanf("%d%d%d",&s,&t,&K);
    if (s==t)
        K++;
    dijkstra(t-1);
    printf("%d\n",a_star(s-1,t-1));
}
```

6.17 稳定婚姻

假定有n个男生和M个女生，理想的拍拖状态就是对于每对情侣(a,b)，找不到另一对情侣(c,d)使得c更喜欢b,b也更喜欢c，同理，对a来说也没有(e,f)使得a更喜欢e而e更喜欢a,当然

最后会有一些人落单。这样子一个状态可以称为理想拍拖状态，它也有一个专业的名词叫稳定婚姻。
求解这个问题可以用一个专有的算法，延迟认可算法，其核心就是让每个男生按自己喜欢的顺序逐个向女生表白，例如leokan向一个女生求爱，这个过程中，若这个女生没有男朋友，那么这个女生就暂时成为leokan的女朋友，或这个女生喜欢她现有男朋友的程度没有喜欢leokan高，这个女生也暂时成为leokan的女朋友，而她原有的男朋友则再将就找下一个次喜欢的女生来当女朋友。

```
#include<string.h>
#include<stdio.h>
#define N 1050
int boy[N][N];
int girl[N][N];
int ans[N];
int cur[N];
int n;
void getMarry(int g)
{
  for (int i=ans[g]+1;i<n;i++)
  {
    int b=girl[g][i]-1;
    if (cur[b]<0)
    {
      ans[g]=i;
      cur[b]=g;
      return;
    }
    int og=cur[b];
    if (boy[b][og] > boy[b][g])
    {
      cur[b]=g;
      ans[g]=i;
      getMarry(og);
      return;
    }
  }
};
int main()
{
  int t,a;
  scanf("%d",&t);
  while(t--)
  {
    memset(girl,0,sizeof(girl));
    memset(boy,0,sizeof(boy));
    scanf("%d",&n);
    for (int i=0;i<n;i++)
      for (int j=0;j<n;j++)
        scanf("%d",&girl[i][j]);
    for (int i=0;i<n;i++)
      for (int j=0;j<n;j++)
      {
        scanf("%d",&a);
        boy[i][a-1]=j;
      }
    memset(cur,0xff,sizeof(cur));
    memset(ans,0xff,sizeof(ans));
    for (int i=0;i<n;i++)
      getMarry(i);
    for (int i=0;i<n;i++)
      printf("%d\n",girl[i][ans[i]]);
  }
  return 0;
}
```

6.18 最小树形图

```
const int inf = 19921005;
int n,m,u,v,cost,dis[1001][1001],L;

void init(int n)
{
    L = 0;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            dis[i][j] = inf;
}

struct Edge
{
    int u,v,cost;
};

Edge e[1001*1001];

int pre[1001],id[1001],visit[1001],in[1001];

int zhuliu(int root,int n,int m,Edge e[])
{
    int res = 0,u,v;
    while (true)
    {
        for (int i = 0; i < n; i++)
            in[i] = inf;
        for (int i = 0; i < m; i++)
            if (e[i].u != e[i].v && e[i].cost < in[e[i].v])
            {
                pre[e[i].v] = e[i].u;
                in[e[i].v] = e[i].cost;
            }
        for (int i = 0; i < n; i++)
            if (i != root)
                if (in[i] == inf)   return -1;
        int tn = 0;
        memset(id,-1,sizeof(id));
        memset(visit,-1,sizeof(visit));
        in[root] = 0;
        for (int i = 0; i < n; i++)
        {
            res += in[i];
            v = i;
            while (visit[v] != i && id[v] == -1 && v != root)
            {
                visit[v] = i;
                v = pre[v];
            }
            if(v != root && id[v] == -1)
            {
                for(int u = pre[v] ; u != v ; u = pre[u])
                    id[u] = tn;
                id[v] = tn++;
            }
        }
        if(tn == 0)    break;
        for (int i = 0; i < n; i++)
            if (id[i] == -1)
                id[i] = tn++;
        for (int i = 0; i < m;)
        {
            int v = e[i].v;
            e[i].u = id[e[i].u];
            e[i].v = id[e[i].v];
            if (e[i].u != e[i].v)
                e[i++].cost -= in[v];
            else
                swap(e[i],e[--m]);
        }
        n = tn;
        root = id[root];
    }
    return res;
}

int main()
{
    freopen("in.txt","r",stdin);
    while (scanf("%d%d",&n,&m) != EOF)
    {
        init(n);
        for (int i = 0; i < m; i++)
        {
            scanf("%d%d%d",&u,&v,&cost);
            if (u == v) continue;
            dis[u][v] = min(dis[u][v],cost);
        }
        L = 0;
        for (int i = 0; i < n; i++)
```

```
            for (int j = 0; j < n; j++)
                if (dis[i][j] != inf)
                {
                    e[L].u = i;
                    e[L].v = j;
                    e[L++].cost = dis[i][j];
                }
        printf("%d\n",zhuliu(0,n,L,e));
    }
    return 0;
}
```

7 计算几何

7.1 基本函数

7.1.1 Point定义

```
struct Point
{
    double x, y;
    Point() {}
    Point(double _x, double _y)
    {
        x = _x, y = _y;
    }
    Point operator -(const Point &b)const
    {
        return Point(x - b.x, y - b.y);
    }
    double operator *(const Point &b)const
    {
        return x * b.y - y * b.x;
    }
    double operator &(const Point &b)const
    {
        return x * b.x + y * b.y;
    }
    void transXY(double B)
    {
        double tx = x,ty = y;
        x = tx*cos(B) - ty*sin(B);
        y = tx*sin(B) + ty*cos(B);
    }
};
```

7.1.2 Line定义

```
struct Line
{
    Point s, e;
    double k;
    Line() {}
    Line(Point _s, Point _e)
    {
        s = _s, e = _e;
        k = atan2(e.y - s.y, e.x - s.x);
    }
    Point operator &(const Line &b)const
    {
        Point res = s;
        //注意：有些题目可能会有直线相交或者重合情况
        //可以把返回值改成pair<Point,int>来返回两直线的状态。
        double t = ((s - b.s) * (b.s - b.e)) / ((s - e) * (b.s -
           b.e));
        res.x += (e.x - s.x) * t;
        res.y += (e.y - s.y) * t;
        return res;
    }
};
```

7.1.3 距离：两点距离

```
double dist2(Point a, Point b)
{
    return (a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y);
}
```

7.1.4 距离：点到直线距离

result:点到直线最近点

```
Point NPT(Point P, Line L)
{
    Point result;
    double a, b, t;

    a = L.p2.x - L.p1.x;
    b = L.p2.y - L.p1.y;
    t = ( (P.x - L.p1.x) * a + (P.y - L.p1.y) * b ) / (a * a + b
       * b);

    result.x = L.p1.x + a * t;
    result.y = L.p1.y + b * t;
    return dist2(P, result);
}
```

7.1.5 距离：点到线段距离

res: 点到线段最近点

```
double dist2(Point p1, Point p2, Point p)
{
    Point res;
    double a, b, t;
    a = p2.x - p1.x;
    b = p2.y - p1.y;
    t = ((p.x - p1.x) * a + (p.y - p1.y) * b) / (a * a + b * b);
    if (t >= 0 && t <= 1)
    {
        res.x = p1.x + a * t;
        res.y = p1.y + b * t;
    }
    else
    {
        if (dist2(p, p1) < dist2(p, p2))
            res = p1;
        else
            res = p2;
    }
    return dist2(p, res);
}
```

旧版

```
double CalcDis(Point a,Point s,Point e) //点到线段距离
{
    if (pmult(Point(s,e),Point(s,a)) < 0 || pmult(Point(e,s),
       Point(e,a)) < 0)
        return min(CalcDis(a,s),CalcDis(a,e));
    return abs(xmult(Point(a,s),Point(a,e)))/CalcDis(s,e);
}
```

7.1.6 面积：多边形

点按逆时针排序。

```
double CalcArea(Point p[], int n)
{
    double res = 0;
    for (int i = 0; i < n; i++)
        res += (p[i] * p[(i + 1) % n]) / 2;
    return res;
}
```

7.1.7 判断：线段相交

```
bool inter(Line l1,Line l2)
{
    return (max(l1.s.x,l1.e.x) >= min(l2.s.x,l2.e.x) &&
            max(l2.s.x,l2.e.x) >= min(l1.s.x,l1.e.x) &&
            max(l1.s.y,l1.e.y) >= min(l2.s.y,l2.e.y) &&
            max(l2.s.y,l2.e.y) >= min(l1.s.y,l1.e.y) &&
            ((l2.s-l1.s)*(l1.e-l1.s))*((l2.e-l1.s)*(l1.e-l1.s))
               <= 0 &&
            ((l1.s-l2.s)*(l2.e-l2.s))*((l1.e-l2.s)*(l2.e-l2.s))
               <= 0);
}
```

7.2 圆

7.2.1 面积：两圆相交

圆不可包含

```
double dis(int x,int y)
{
    return sqrt((double)(x*x+y*y));
}
double area(int x1,int y1,int x2,int y2,double r1,double r2)
{
    double s=dis(x2-x1,y2-y1);
    if(r1+r2<s) return 0;
    else if(r2-r1>s) return PI*r1*r1;
    else if(r1-r2>s) return PI*r2*r2;
    double q1=acos((r1*r1+s*s-r2*r2)/(2*r1*s));
    double q2=acos((r2*r2+s*s-r1*r1)/(2*r2*s));
    return (r1*r1*q1+r2*r2*q2-r1*s*sin(q1));
}
```

7.2.2 三角形外接圆

```
void CircumscribedCircle()
{
    for (int i = 0; i < 3; i++)
        scanf("%lf%lf",&p[i].x,&p[i].y);
    tp = Point((p[0].x+p[1].x)/2,(p[0].y+p[1].y)/2);
    l[0] = Line(tp,Point(tp.x-(p[1].y-p[0].y),tp.y+(p[1].x-p[0].x
       )));
    tp = Point((p[0].x+p[2].x)/2,(p[0].y+p[2].y)/2);
    l[1] = Line(tp,Point(tp.x-(p[2].y-p[0].y),tp.y+(p[2].x-p[0].x
       )));
    tp = LineToLine(l[0],l[1]);
    r = Point(tp,p[0]).Length();
    printf("(%.6f,%.6f,%.6f)\n",tp.x,tp.y,r);
}
```

7.2.3 三角形内切圆

```
void InscribedCircle()
{
    for (int i = 0; i < 3; i++)
        scanf("%lf%lf",&p[i].x,&p[i].y);
    if (xmult(Point(p[0],p[1]),Point(p[0],p[2])) < 0)
        swap(p[1],p[2]);
    for (int i = 0; i < 3; i++)
        len[i] = Point(p[i],p[(i+1)%3]).Length();
    tr = (len[0]+len[1]+len[2])/2;
    r = sqrt((tr-len[0])*(tr-len[1])*(tr-len[2])/tr);
    for (int i = 0; i < 2; i++)
    {
        v = Point(p[i],p[i+1]);
        tv = Point(-v.y,v.x);
        tr = tv.Length();
        tv = Point(tv.x*r/tr,tv.y*r/tr);
        tp = Point(p[i].x+tv.x,p[i].y+tv.y);
        l[i].s = tp;
        tp = Point(p[i+1].x+tv.x,p[i+1].y+tv.y);
        l[i].e = tp;
    }
    tp = LineToLine(l[0],l[1]);
    printf("(%.6f,%.6f,%.6f)\n",tp.x,tp.y,r);
}
```

7.2.4 点对圆的两个切点

```
void calc_qie(Point poi,Point o,double r,Point &result1,Point &
   result2) {
    double line=sqrt((poi.x-o.x)*(poi.x-o.x)+(poi.y-o.y)*(poi.y-o
       .y));
    double angle=acos(r/line);
    Point unitvector,lin;
    lin.x=poi.x-o.x;
    lin.y=poi.y-o.y;
    unitvector.x=lin.x/sqrt(lin.x*lin.x+lin.y*lin.y)*r;
    unitvector.y=lin.y/sqrt(lin.x*lin.x+lin.y*lin.y)*r;
    result1=Rotate(unitvector,-angle);
    result2=Rotate(unitvector,angle);
    result1.x+=o.x;
    result1.y+=o.y;
    result2.x+=o.x;
    result2.y+=o.y;
    return;
}
```

7.2.5 两圆公切点

```
void Gao()
{
    tn = 0;
    Point a,b,vab;
    double tab,tt,dis,theta;
    for (int i = 0; i < tc; i++)
        for (int j = 0; j < tc; j++)
            if (i != j)
            {
                a = c[i];
                b = c[j];
                vab = Point(a,b);
                tab = atan2(vab.y,vab.x);
                dis = sqrt(vab.x*vab.x+vab.y*vab.y);
                if (b.r > a.r)
                    tt = asin((b.r-a.r)/dis);
                else
                    tt = -asin((a.r-b.r)/dis);
                theta = tab+pi/2+tt;
                tp[tn++] = Point(a.x+a.r*cos(theta),a.y+a.r*sin(
                   theta));
                tp[tn++] = Point(b.x+b.r*cos(theta),b.y+b.r*sin(
                   theta));
            }
}
```

7.3 矩阵

7.3.1 基本矩阵

按向量$\overrightarrow{(x,y,z)}$平移:

$$\begin{pmatrix}1 & 0 & 0 & x\\ 0 & 1 & 0 & y\\ 0 & 0 & 1 & z\\ 0 & 0 & 0 & 1\end{pmatrix}$$

按比例(x, y, z)缩放：

$$\begin{pmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

绕向量$\overrightarrow{(x, y, z)}$旋转$angle$角度：

$$\begin{pmatrix} x^2 \times (1-c) + c & x \times y \times (1-c) - z \times s & x \times z \times (1-c) + y \times s & 0 \\ y \times x \times (1-c) + z \times s & y^2 \times (1-c) + c & y \times z \times (1-c) - x \times s & 0 \\ x \times z \times (1-c) - y \times s & y \times z \times (1-c) + x \times s & z^2 \times (1-c) + c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{cases} s = sin(angle) \\ c = cos(angle) \end{cases}$$

7.3.2 刘汝佳的几何教室

```
const double pi = acos(-1.0);

int n,m,q;
struct Point
{
    double a,b,c,d;
};
Point p[50000],f[50000];

double a,b,c,theta,mt[4][4],tmp[4][4],tmt[4][4],rmt[4][8];
char com[20];

void TRANSLATE()
{
    memset(tmt,0,sizeof(tmt));
    tmt[0][0] = tmt[1][1] = tmt[2][2] = tmt[3][3] = 1;
    tmt[3][0] = a;
    tmt[3][1] = b;
    tmt[3][2] = c;
    memset(tmp,0,sizeof(tmp));
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
            for (int k = 0; k < 4; k++)
                tmp[i][j] += mt[i][k]*tmt[k][j];
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
            mt[i][j] = tmp[i][j];
}

void ROTATE()
{
    theta = -theta*pi/180;
    memset(tmt,0,sizeof(tmt));
    tmt[3][3] = 1;
    tmt[0][0] = cos(theta)+(1-cos(theta))*a*a;
    tmt[1][0] = (1-cos(theta))*a*b+c*sin(theta);
    tmt[2][0] = (1-cos(theta))*a*c-b*sin(theta);
    tmt[0][1] = (1-cos(theta))*a*b-c*sin(theta);
    tmt[1][1] = cos(theta)+(1-cos(theta))*b*b;
    tmt[2][1] = (1-cos(theta))*b*c+a*sin(theta);
    tmt[0][2] = (1-cos(theta))*a*c+b*sin(theta);
    tmt[1][2] = (1-cos(theta))*b*c-a*sin(theta);
    tmt[2][2] = cos(theta)+(1-cos(theta))*c*c;
    memset(tmp,0,sizeof(tmp));
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
            for (int k = 0; k < 4; k++)
                tmp[i][j] += mt[i][k]*tmt[k][j];
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
            mt[i][j] = tmp[i][j];
}

void SCALE()
{
    memset(tmt,0,sizeof(tmt));
    tmt[0][0] = a;
    tmt[1][1] = b;
    tmt[2][2] = c;
    tmt[3][3] = 1;
    memset(tmp,0,sizeof(tmp));
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
            for (int k = 0; k < 4; k++)
                tmp[i][j] += mt[i][k]*tmt[k][j];
    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
            mt[i][j] = tmp[i][j];
}

void solvep(Point p)
{
    memset(tmt,0,sizeof(tmt));
    tmt[0][0] = p.a;
    tmt[0][1] = p.b;
    tmt[0][2] = p.c;
    tmt[0][3] = 1;
    memset(tmp,0,sizeof(tmp));
    for (int i = 0; i < 1; i++)
        for (int j = 0; j < 4; j++)
            for (int k = 0; k < 4; k++)
                tmp[i][j] += tmt[i][k]*mt[k][j];
    printf("%.2f␣%.2f␣%.2f\n",tmp[0][0],tmp[0][1],tmp[0][2]);
}

void solvef(Point f)
{
    memset(tmt,0,sizeof(tmt));
    tmt[0][0] = f.a;
    tmt[1][0] = f.b;
    tmt[2][0] = f.c;
    tmt[3][0] = 0;
    memset(tmp,0,sizeof(tmp));
    for (int i = 0;i < 4;i++)
        for (int j = 0;j < 1;j++)
            for (int k = 0;k < 4;k++)
                tmp[i][j] += mt[i][k]*tmt[k][j];
    tmp[3][0] += f.d;
    double kk = tmp[0][0]*tmp[0][0]+tmp[1][0]*tmp[1][0]+tmp
       [2][0]*tmp[2][0];
    kk = sqrt(1/kk);
    for (int i = 0;i < 4;i++)
        printf("%.2f␣",tmp[i][0]*kk);
    printf("\n");
}

void solvermt()
{
    memset(rmt,0,sizeof(rmt));
    for (int i = 0;i < 4;i++)
        for (int j = 0;j < 4;j++)
            rmt[i][j] = mt[i][j];
    rmt[0][4] = rmt[1][5] = rmt[2][6] = rmt[3][7] = 1;
    for (int i = 0;i < 4;i++)
    {
        for (int j = i;j < 4;j++)
            if (fabs(rmt[j][i]) > 1e-8)
            {
                for (int k = i;k < 8;k++)
                    swap(rmt[i][k],rmt[j][k]);
                break;
            }
        double tt = rmt[i][i];
        for (int j = i;j < 8;j++)
            rmt[i][j] /= tt;
        for (int j = 0;j < 4;j++)
            if (i != j)
            {
                tt = rmt[j][i];
                for (int k = i;k < 8;k++)
                    rmt[j][k] -= rmt[i][k]*tt;
            }
    }
    for (int i = 0;i < 4;i++)
        for (int j = 0;j < 4;j++)
            mt[i][j] = rmt[i][4+j];
}

int main()
```

```
{
    scanf("%d%d%d",&n,&m,&q);
    for (int i = 0; i < n; i++)
        scanf("%lf%lf%lf",&p[i].a,&p[i].b,&p[i].c);
    for (int i = 0; i < m; i++)
        scanf("%lf%lf%lf%lf",&f[i].a,&f[i].b,&f[i].c,&f[i].d);
    memset(mt,0,sizeof(mt));
    mt[0][0] = mt[1][1] = mt[2][2] = mt[3][3] = 1;
    for (int i = 0; i < q; i++)
    {
        scanf("%s",com);
        if (strcmp(com,"TRANSLATE") == 0)
        {
            scanf("%lf%lf%lf",&a,&b,&c);
            TRANSLATE();
        }
        else if (strcmp(com,"ROTATE") == 0)
        {
            scanf("%lf%lf%lf%lf",&a,&b,&c,&theta);
            ROTATE();
        }
        else if (strcmp(com,"SCALE") == 0)
        {
            scanf("%lf%lf%lf",&a,&b,&c);
            SCALE();
        }
    }
  //处理点
    for (int i = 0; i < n; i++)
        solvep(p[i]);
  //处理面
    solvermt();
    for (int i = 0; i < m; i++)
        solvef(f[i]);
    return 0;
}
```

7.4 重心

```
Point CenterOfPolygon(Point poly[],int n)
{
    Point p, p0, p1, p2, p3;
    double m, m0;
    p1 = poly[0];
    p2 = poly[1];
    p.x = p.y = m = 0;
    for (int i = 2; i < n; i++)
    {
  p3 = poly[i];
  p0.x = (p1.x + p2.x + p3.x) / 3.0;
  p0.y = (p1.y + p2.y + p3.y) / 3.0;
  m0 = p1.x * p2.y + p2.x * p3.y + p3.x * p1.y - p1.y * p2.x - p2.y * p3.x - p3.y * p1.x;
  if (cmp(m + m0,0.0) == 0)
      m0 += eps;
  p.x = (m * p.x + m0 * p0.x) / (m + m0);
  p.y = (m * p.y + m0 * p0.y) / (m + m0);
  m = m + m0;
  p2 = p3;
    }
    return p;
}
```

7.5 KD树

查找某个点距离最近的点，基本思想是每次分治把点分成两部分，建议按照坐标规模决定是垂直划分还是水平划分，查找时先往分到的那一部分查找，然后根据当前最优答案决定是否去另一个区间查找。

```
bool Div[MaxN];
void BuildKD(int deep,int l, int r, Point p[])\\记得备份一下P
{
    if (l > r) return;
    int mid = l + r >> 1;
    int minX, minY, maxX, maxY;
    minX = min_element(p + l, p + r + 1, cmpX)->x;
    minY = min_element(p + l, p + r + 1, cmpY)->y;
    maxX = max_element(p + l, p + r + 1, cmpX)->x;
    maxY = max_element(p + l, p + r + 1, cmpY)->y;
    Div[mid] = (maxX - minX >= maxY - minY);
    nth_element(p + l, p + mid, p + r + 1, Div[mid] ? cmpX : cmpY
       );
    BuildKD(l, mid - 1, p);
    BuildKD(mid + 1, r, p);
}

long long res;
void Find(int l, int r, Point a, Point p[])\\查找
{
    if (l > r)  return;
    int mid = l + r >> 1;
    long long dist = dist2(a, p[mid]);
    if (dist > 0)//如果有重点不能这样判断
        res = min(res, dist);
    long long d = Div[mid] ? (a.x - p[mid].x) : (a.y - p[mid].y);
    int l1, l2, r1, r2;
    l1 = l, l2 = mid + 1;
    r1 = mid - 1, r2 = r;
    if (d > 0)
        swap(l1, l2), swap(r1, r2);
    Find(l1, r1, a, p);
    if (d * d < res)
        Find(l2, r2, a, p);
}
```

7.5.1 例题

查询一个点为中心的给定正方形内所有点并删除（2012金华网赛A）

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>
#include <queue>
using namespace std;

const int MaxN = 100000;
struct Point
{
  int x,y,r;
  int id;
  bool del;
};

int cmpTyp;
bool cmp(const Point& a,const Point& b)
{
  if (cmpTyp == 0)
    return a.x < b.x;
  else
    return a.y < b.y;
}

int cnt[MaxN];
bool Div[MaxN];
int minX[MaxN],minY[MaxN],maxX[MaxN],maxY[MaxN];
void BuildKD(int l,int r,Point p[])
{
  if (l > r)  return;
  int mid = l+r>>1;
  cmpTyp = 0;
  minX[mid] = min_element(p+l,p+r+1,cmp)->x;
  maxX[mid] = max_element(p+l,p+r+1,cmp)->x;
  cmpTyp = 1;
  minY[mid] = min_element(p+l,p+r+1,cmp)->y;
  maxY[mid] = max_element(p+l,p+r+1,cmp)->y;

  cnt[mid] = r-l+1;
  cmpTyp = Div[mid] = (maxX[mid]-minX[mid] < maxY[mid]-minY[mid])
      ;
  nth_element(p+l,p+mid,p+r+1,cmp);
  BuildKD(l,mid-1,p);
  BuildKD(mid+1,r,p);
}

queue<int> Q;
```

```
int Find(int l,int r,Point a,Point p[])
{
  if (l > r)  return 0;
  int mid = l+r>>1;
  if (cnt[mid] == 0)  return 0;

  if (maxX[mid] < a.x-a.r ||
      minX[mid] > a.x+a.r ||
      maxY[mid] < a.y-a.r ||
      minY[mid] > a.y+a.r)
    return 0;

  int totdel = 0;

  if (p[mid].del == false)
    if (abs(p[mid].x-a.x) <= a.r && abs(p[mid].y-a.y) <= a.r)
    {
      p[mid].del = true;
      Q.push(p[mid].id);
      totdel++;
    }

  totdel += Find(l,mid-1,a,p);
  totdel += Find(mid+1,r,a,p);

  cnt[mid] -= totdel;

  return totdel;
}

Point p[MaxN],tp[MaxN];
int n;

int main()
{
  int cas = 1;
  while (true)
  {
    scanf("%d",&n);
    if (n == 0) break;

    for (int i = 0;i < n;i++)
    {
      p[i].id = i;
      int tx,ty;
      scanf("%d%d%d",&tx,&ty,&p[i].r);
      p[i].x = tx-ty;
      p[i].y = tx+ty;
      p[i].del = false;
      tp[i] = p[i];
    }
    BuildKD(0,n-1,tp);

    printf("Case␣#%d:\n",cas++);
    int q;
    scanf("%d",&q);
    for (int i = 0;i < q;i++)
    {
      int id;
      scanf("%d",&id);
      int res = 0;
      id--;
      Q.push(id);
      while (!Q.empty())
      {
        int now = Q.front();
        Q.pop();
        if (p[now].del == true) continue;
        p[now].del = true;
        res += Find(0,n-1,p[now],tp);
      }
      printf("%d\n",res);
    }
  }
  return 0;
}
```

7.6 半平面交

直线左边代表有效区域。

```
bool HPIcmp(Line a, Line b)
{
    if (fabs(a.k - b.k) > eps)    return a.k < b.k;
    return ((a.s - b.s) * (b.e-b.s)) < 0;
}

Line Q[100];
void HPI(Line line[], int n, Point res[], int &resn)
{
    int tot = n;
    sort(line, line + n, HPIcmp);
    tot = 1;
    for (int i = 1; i < n; i++)
        if (fabs(line[i].k - line[i - 1].k) > eps)
            line[tot++] = line[i];
    int head = 0, tail = 1;
    Q[0] = line[0];
    Q[1] = line[1];
    resn = 0;
    for (int i = 2; i < tot; i++)
    {
        if (fabs((Q[tail].e-Q[tail].s) * (Q[tail - 1].e-Q[tail -
           1].s)) < eps ||
                fabs((Q[head].e-Q[head].s) * (Q[head + 1].e-Q[
                    head + 1].s)) < eps)
            return;
        while (head < tail && (((Q[tail]&Q[tail - 1]) - line[i].s
            ) * (line[i].e-line[i].s)) > eps)
            tail--;
        while (head < tail && (((Q[head]&Q[head + 1]) - line[i].s
            ) * (line[i].e-line[i].s)) > eps)
            head++;
        Q[++tail] = line[i];
    }
    while (head < tail && (((Q[tail]&Q[tail - 1]) - Q[head].s) *
        (Q[head].e-Q[head].s)) > eps)
        tail--;
    while (head < tail && (((Q[head]&Q[head + 1]) - Q[tail].s) *
        (Q[tail].e-Q[tail].s)) > eps)
        head++;
    if (tail <= head + 1) return;
    for (int i = head; i < tail; i++)
        res[resn++] = Q[i] & Q[i + 1];
    if (head < tail + 1)
        res[resn++] = Q[head] & Q[tail];
}
```

7.7 凸包

得到的凸包按照逆时针方向排序。

```
bool GScmp(Point a, Point b)
{
    if (fabs(a.x - b.x) < eps)
        return a.y < b.y - eps;
    return a.x < b.x - eps;
}

void GS(Point p[], int n, Point res[], int &resn)
{
    resn = 0;
    int top = 0;
    sort(p, p + n, GScmp);
    for (int i = 0; i < n;)
        if (resn < 2 || (res[resn - 1] - res[resn - 2]) * (p[i] -
            res[resn - 1]) > eps)
            res[resn++] = p[i++];
        else
            --resn;
    top = resn - 1;
    for (int i = n - 2; i >= 0;)
        if (resn < top + 2 || (res[resn - 1] - res[resn - 2]) * (
            p[i] - res[resn - 1]) > eps)
            res[resn++] = p[i--];
        else
            --resn;
    resn--;
    if (resn < 3)   resn = 0;
}
```

7.8 直线与凸包求交点

复杂度$O(\log n)$。
需要先预处理几个东西。

```
//二分[la,lb]这段区间那条边与line相交
int Gao(int la,int lb,Line line)
{
    if (la > lb)
        lb += n;
    int l = la,r = lb,mid;
    while (l < r)
    {
        mid = l+r+1>>1;
        if (cmp((line.e-line.s)*(p[la]-line.s),0)*cmp((line.e-
           line.s)*(p[mid]-line.s),0) >= 0)
            l = mid;
        else
            r = mid-1;
    }
    return l%n;
}
//求l与凸包的交点

//先调用Gettheta预处理出凸包每条边的斜率，然后处理成升序排列
double theta[maxn];

void Gettheta()
{
    for (int i = 0;i < n;i++)
    {
        Point v = p[(i+1)%n]-p[i];
        theta[i] = atan2(v.y,v.x);
    }
    for (int i = 1;i < n;i++)
        if (theta[i-1] > theta[i]+eps)
            theta[i] += 2*pi;
}

double Calc(Line l)
{
    double tnow;
    Point v = l.e-l.s;
    tnow = atan2(v.y,v.x);
    if (cmp(tnow,theta[0]) < 0)    tnow += 2*pi;
    int pl = lower_bound(theta,theta+n,tnow)-theta;
    tnow = atan2(-v.y,-v.x);
    if (cmp(tnow,theta[0]) < 0)    tnow += 2*pi;
    int pr = lower_bound(theta,theta+n,tnow)-theta;
    //pl和pr是在l方向上距离最远的点对
    pl = pl%n;
    pr = pr%n;

    if (cmp(v*(p[pl]-l.s),0)*cmp(v*(p[pr]-l.s),0) >= 0)
        return 0.0;

    int xa = Gao(pl,pr,l);
    int xb = Gao(pr,pl,l);

    if (xa > xb)    swap(xa,xb);
    //与[xa,xa+1]和[xb,xb+1]这两条线段相交

    if (cmp(v*(p[xa+1]-p[xa]),0) == 0)  return 0.0;
    if (cmp(v*(p[xb+1]-p[xb]),0) == 0)  return 0.0;

    Point pa,pb;
    pa = Line(p[xa],p[xa+1])&l;
    pb = Line(p[xb],p[xb+1])&l;
    //题目：求直线切凸包得到的两部分的面积
    double area0 = sum[xb]-sum[xa+1]+(pa*p[xa+1])/2.0+(p[xb]*pb)
       /2.0+(pb*pa)/2.0;
    double area1 = sum[xa+n]-sum[xb+1]+(pb*p[xb+1])/2.0+(p[xa]*pa
       )/2.0+(pa*pb)/2.0;

    return min(area0,area1);
}
```

7.9 三维凸包

暴力写法

```
#define eps 1e-7
#define MAXV 505

struct pt
{
    double x, y, z;
    pt() {}
    pt(double _x, double _y, double _z): x(_x), y(_y), z(_z) {}
    pt operator - (const pt p1)
    {
        return pt(x - p1.x, y - p1.y, z - p1.z);
    }
    pt operator * (pt p)
    {
        return pt(y*p.z-z*p.y, z*p.x-x*p.z, x*p.y-y*p.x);
    }
    double operator ^ (pt p)
    {
        return x*p.x+y*p.y+z*p.z;
    }
};
struct _3DCH
{
    struct fac
    {
        int a, b, c;
        bool ok;
    };
    int n;
    pt P[MAXV];
    int cnt;
    fac F[MAXV*8];
    int to[MAXV][MAXV];
    double vlen(pt a)
    {
        return sqrt(a.x*a.x+a.y*a.y+a.z*a.z);
    }
    double area(pt a, pt b, pt c)
    {
        return vlen((b-a)*(c-a));
    }
    double volume(pt a, pt b, pt c, pt d)
    {
        return (b-a)*(c-a)^(d-a);
    }
    double ptof(pt &p, fac &f)
    {
        pt m = P[f.b]-P[f.a], n = P[f.c]-P[f.a], t = p-P[f.a];
        return (m * n) ^ t;
    }
    void deal(int p, int a, int b)
    {
        int f = to[a][b];
        fac add;
        if (F[f].ok)
        {
            if (ptof(P[p], F[f]) > eps)
                dfs(p, f);
            else
            {
                add.a = b, add.b = a, add.c = p, add.ok = 1;
                to[p][b] = to[a][p] = to[b][a] = cnt;
                F[cnt++] = add;
            }
        }
    }
    void dfs(int p, int cur)
    {
        F[cur].ok = 0;
        deal(p, F[cur].b, F[cur].a);
        deal(p, F[cur].c, F[cur].b);
        deal(p, F[cur].a, F[cur].c);
    }
    bool same(int s, int t)
    {
        pt &a = P[F[s].a], &b = P[F[s].b], &c = P[F[s].c];
        return fabs(volume(a, b, c, P[F[t].a])) < eps && fabs(volume(a, b, c,
                P[F[t].b])) < eps && fabs(volume(a, b, c, P[F[t].c])) < eps;
    }
    void construct()
    {
        cnt = 0;
        if (n < 4)
            return;
        bool sb = 1;
        for (int i = 1; i < n; i++)
        {
            if (vlen(P[0] - P[i]) > eps)
            {
                swap(P[1], P[i]);
                sb = 0;
                break;
            }
        }
        if (sb)return;
        sb = 1;
        for (int i = 2; i < n; i++)
        {
            if (vlen((P[0] - P[1]) * (P[1] - P[i])) > eps)
            {
                swap(P[2], P[i]);
                sb = 0;
                break;
            }
        }
        if (sb)return;
        sb = 1;
        for (int i = 3; i < n; i++)
        {
            if (fabs((P[0] - P[1]) * (P[1] - P[2]) ^ (P[0] - P[i])) > eps)
            {
                swap(P[3], P[i]);
                sb = 0;
                break;
            }
        }
        if (sb)return;
        fac add;
        for (int i = 0; i < 4; i++)
        {
                add.a = (i+1)%4, add.b = (i+2)%4, add.c = (i+3)%4,
                   add.ok = 1;
                if (ptof(P[i], add) > 0)
                    swap(add.b, add.c);
                to[add.a][add.b] = to[add.b][add.c] = to[add.c][add.a
                   ] = cnt;
                F[cnt++] = add;
            }
            for (int i = 4; i < n; i++)
            {
                for (int j = 0; j < cnt; j++)
                {
                    if (F[j].ok && ptof(P[i], F[j]) > eps)
                    {
                        dfs(i, j);
                        break;
                    }
                }
            }
            int tmp = cnt;
            cnt = 0;
            for (int i = 0; i < tmp; i++)
            {
                if (F[i].ok)
                {
                    F[cnt++] = F[i];
                }
            }
        }
//表面积
        double area()
        {
            double ret = 0.0;
            for (int i = 0; i < cnt; i++)
            {
                ret += area(P[F[i].a], P[F[i].b], P[F[i].c]);
            }
            return ret / 2.0;
        }
//体积
        double volume()
        {
            pt O(0, 0, 0);
            double ret = 0.0;
            for (int i = 0; i < cnt; i++)
            {
                ret += volume(O, P[F[i].a], P[F[i].b], P[F[i].c]);
            }
            return fabs(ret / 6.0);
        }
//表面三角形数
        int facetCnt_tri()
```

```
    {
        return cnt;
    }
//表面多边形数
    int facetCnt()
    {
        int ans = 0;
        for (int i = 0; i < cnt; i++)
        {
            bool nb = 1;
            for (int j = 0; j < i; j++)
            {
                if (same(i, j))
                {
                    nb = 0;
                    break;
                }
            }
            ans += nb;
        }
        return ans;
    }

    pt Fc[MAXV*8];
    double V[MAXV*8];
    pt Center()//重心
    {
        pt O(0,0,0);
        for (int i = 0; i < cnt; i++)
        {
            Fc[i].x = (O.x+P[F[i].a].x+P[F[i].b].x+P[F[i].c].x)
               /4.0;
            Fc[i].y = (O.y+P[F[i].a].y+P[F[i].b].y+P[F[i].c].y)
               /4.0;
            Fc[i].z = (O.z+P[F[i].a].z+P[F[i].b].z+P[F[i].c].z)
               /4.0;
            V[i] = volume(O,P[F[i].a],P[F[i].b],P[F[i].c]);
        }
        pt res = Fc[0],tmp;
        double m = V[0];
        for (int i = 1; i < cnt; i++)
        {
            if (fabs(m+V[i]) < eps)
                V[i] += eps;
            tmp.x = (m*res.x+V[i]*Fc[i].x)/(m+V[i]);
            tmp.y = (m*res.y+V[i]*Fc[i].y)/(m+V[i]);
            tmp.z = (m*res.z+V[i]*Fc[i].z)/(m+V[i]);
            m += V[i];
            res = tmp;
        }
        return res;
    }
```

```
};

_3DCH hull;

int main()
{
    while (scanf("%d",&hull.n) != EOF)
    {
        for (int i = 0; i < hull.n; i++)
            scanf("%lf%lf%lf",&hull.P[i].x,&hull.P[i].y,&hull.P[i
              ].z);
        hull.construct();
    }
    return 0;
}
```

7.10 旋转卡壳

“对踵”

7.10.1 单个凸包

```
void solve(Point p[],int n)
{
    Point v;
    int cur = 1;
    for (int i = 0;i < n;i++)
    {
        v = p[i]-p[(i+1)%n];
        while (v*(p[(cur+1)%n]-p[cur]) < 0)
            cur = (cur+1)%n;
        //p[cur] -> p[i]
        //p[cur] -> p[i+1]
        //p[cur] -> (p[i],p[i+1])
    }
}
```

7.10.2 两个凸包

注意初始点的选取，代码只是个示例。
有时候答案需要取solve(p0,n,p1,m)和solve(p1,m,p0,n)的最优值。
何老鱼说我的是错的。。

```
void solve(Point p0[],int n,Point p1[],int m)
{
    Point v;
    int cur = 0;
    for (int i = 0;i < n;i++)
    {
        v = p0[i]-p0[(i+1)%n];
        while (v*(p1[(cur+1)%m]-p1[cur]) < 0)
            cur = (cur+1)%m;
        //p1[cur] -> p0[i]
        //p1[cur] -> p0[i+1]
        //p1[cur] -> (p0[i],p0[i+1])
    }
}
```

7.10.3 外接矩形

```
void solve()
{
    resa = resb = 1e100;
    double dis1,dis2;
    Point xp[4];
    Line l[4];
    int a,b,c,d;
    int sa,sb,sc,sd;
    a = b = c = d = 0;
    sa = sb = sc = sd = 0;
    Point va,vb,vc,vd;
    for (a = 0; a < n; a++)
    {
        va = Point(p[a],p[(a+1)%n]);
        vc = Point(-va.x,-va.y);
        vb = Point(-va.y,va.x);
        vd = Point(-vb.x,-vb.y);
        if (sb < sa)
        {
            b = a;
            sb = sa;
        }
        while (xmult(vb,Point(p[b],p[(b+1)%n])) < 0)
        {
            b = (b+1)%n;
            sb++;
        }
        if (sc < sb)
        {
            c = b;
            sc = sb;
        }
        while (xmult(vc,Point(p[c],p[(c+1)%n])) < 0)
        {
            c = (c+1)%n;
            sc++;
        }
        if (sd < sc)
        {
            d = c;
            sd = sc;
        }
        while (xmult(vd,Point(p[d],p[(d+1)%n])) < 0)
        {
                d = (d+1)%n;
                sd++;
            }

            //卡在p[a],p[b],p[c],p[d]上
            sa++;
        }
}
```

7.11 三角形内点个数

7.11.1 无三点共线

```
Point p[1000], tp[2000], base;

bool cmp(const Point &a, const Point &b)
{
  return a.theta < b.theta;
}

int cnt[1000][1000];
int cntleft[1000][1000];
int n, m;

int calc(int a, int b, int c)
{
    Point p1 = p[b] - p[a], p2 = p[c] - p[a];
    if (atan2(p1.y, p1.x) > atan2(p2.y, p2.x))
        swap(b, c);
    if ((p[b] - p[a]) * (p[c] - p[a]) > 0)
        return cnt[a][c] - cnt[a][b] - 1;
    else
        return n - 3 - (cnt[a][c] - cnt[a][b] - 1);
}

int main(int argc, char const *argv[])
{
    int totcas;
    scanf("%d", &totcas);
    for (int cas = 1; cas <= totcas; ++cas)
    {
        scanf("%d", &n);
        for (int i = 0; i < n; ++i)
        {
            scanf("%lld%lld", &p[i].x, &p[i].y);
            p[i].id = i;
        }
        for (int i = 0; i < n; ++i)
        {
            m = 0;
            base = p[i];
            for (int j = 0; j < n; ++j)
                if (i != j)
                {
                    tp[m] = p[j];
                    Point v = tp[m]-base;
                    tp[m++].theta = atan2(v.y,v.x);
                }

            sort(tp, tp + m, cmp);
            for (int j = 0; j < m; ++j)
                tp[m + j] = tp[j];

            //calc cnt
            for (int j = 0; j < m; ++j)
                cnt[i][tp[j].id] = j;

            //calc cntleft
            for (int j = 0, k = 0, tot = 0; j < m; ++j)
            {
                while (k == j || (k < j + m && (tp[j] - base) * (
                   tp[k] - base) > 0))
                    k++, tot++;
                cntleft[i][tp[j].id] = --tot;
            }
        }

        printf("Case␣%d:\n", cas);
        int q;
        scanf("%d", &q);
        for (int i = 0; i < q; ++i)
        {
            int x, y, z;
            scanf("%d%d%d", &x, &y, &z);
            if ((p[z] - p[x]) * (p[y] - p[x]) > 0)
                swap(y, z);
            int res = cntleft[x][z] + cntleft[z][y] + cntleft[y][
               x];
            res += calc(x, y, z) + calc(y, z, x) + calc(z, x, y);
            res -= 2 * (n - 3);
            printf("%d\n", res);
        }
    }
    return 0;
}
```

7.11.2 有三点共线且点有类别之分

```
int n,n0,n1,m;
Point p[3000], tp[3000], base;

bool cmp(const Point &a, const Point &b)
{
    if ((a-base)*(b-base) == 0)
    {
        return (a-base).getMol() < (b-base).getMol();
    }
    return a.theta < b.theta;
}

int cnt[100][100];
int cntleft[100][100];

int calc(int a,int b,int c)
{
    Point p1 = p[b]-p[a],p2 = p[c]-p[a];
    if (atan2(1.0*p1.y,1.0*p1.x) > atan2(1.0*p2.y,1.0*p2.x))
        swap(b,c);
    int res = cnt[a][c]-cnt[a][b];
    if ((p[b]-p[a])*(p[c]-p[a]) > 0)
        return res;
    else
        return n1-res;
}

int main()
{
    int cas = 0;
    while (scanf("%d%d",&n0,&n1) != EOF)
    {
        n = n1+n0;
        for (int i = 0; i < n; i++)
        {
            scanf("%I64d%I64d",&p[i].x,&p[i].y);
            p[i].id = i;
        }
        for (int i = 0; i < n0; ++i)
        {
            m = 0;
            base = p[i];
            for (int j = 0; j < n; ++j)
                if (i != j)
                {
                    tp[m] = p[j];
                    Point v = tp[m]-base;
                    tp[m++].theta = atan2(1.0*v.y,1.0*v.x);
                }

            sort(tp, tp + m, cmp);
            for (int j = 0; j < m; ++j)
                tp[m + j] = tp[j];

            for (int j = 0,tot = 0; j < m; ++j)
            {
                if (tp[j].id < n0)
                    cnt[i][tp[j].id] = tot;
                else
                    tot++;
            }

            for (int j = 0, k = 0, tot = 0; j < m; ++j)
            {
                while (k == j || (k < j + m && (tp[j] - base) * (tp[k] - base) > 0))
                {
                    if (tp[k].id >= n0)
                        tot++;
                    k++;
                }
                if (tp[j].id >= n0)
                    tot--;
                else
                    cntleft[i][tp[j].id] = tot;
            }
        }

        int ans = 0;
        for (int i = 0; i < n0; i++)
            for (int j = i+1; j < n0; j++)
                for (int k = j+1; k < n0; k++)
                {
                    int x = i,y = j,z = k;

                    if ((p[z] - p[x]) * (p[y] - p[x]) > 0)
                        swap(y, z);
                    int res = cntleft[x][z] + cntleft[z][y] + cntleft[y][x];

                    res += calc(x, y, z) + calc(y, z, x) + calc(z, x, y);

                    res -= 2 * n1;

                    //printf("%d %d %d %d\n",x,y,z,res);

                    if (res%2 == 1)
                        ans++;
                }
        printf("Case␣%d:␣%d\n",++cas,ans);
    }
    return 0;
}
```

7.12 最近点对

7.12.1 类快排算法

```
double calc_dis(Point &a ,Point &b) {
    return sqrt((a.x-b.x)*(a.x-b.x) + (a.y-b.y)*(a.y-b.y));
}
//别忘了排序
```

```
bool operator<(const Point &a ,const Point &b) {
    if(a.y != b.y) return a.x < b.x;
    return a.x < b.x;
}
double Gao(int l ,int r ,Point pnts[]) {
    double ret = inf;
    if(l == r) return ret;
    if(l+1 ==r) {
        ret = min(calc_dis(pnts[l],pnts[l+1]) ,ret);
        return ret;
    }
    if(l+2 ==r) {
        ret = min(calc_dis(pnts[l],pnts[l+1]) ,ret);
        ret = min(calc_dis(pnts[l],pnts[l+2]) ,ret);
        ret = min(calc_dis(pnts[l+1],pnts[l+2]) ,ret);
        return ret;
    }

    int mid = l+r>>1;
    ret = min (ret ,Gao(l ,mid,pnts));
    ret = min (ret , Gao(mid+1, r,pnts));

    for(int c = l ; c<=r; c++)
        for(int d = c+1; d <=c+7 && d<=r; d++) {
            ret = min(ret , calc_dis(pnts[c],pnts[d]));
        }
    return ret;
}
```

7.12.2 随机增量法

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <map>
#include <vector>
#include <cmath>
#include <algorithm>
#define Point pair<double,double>
using namespace std;

const int step[9][2] =
   {{-1,-1},{-1,0},{-1,1},{0,-1},{0,0},{0,1},{1,-1},{1,0},{1,1}};
int n,x,y,nx,ny;
map<pair<int,int>,vector<Point > > g;
vector<Point > tmp;
Point p[20000];
double tx,ty,ans,nowans;
vector<Point >::iterator it,op,ed;
pair<int,int> gird;
bool flag;

double Dis(Point p0,Point p1)
```

```
{
    return sqrt((p0.first-p1.first)*(p0.first-p1.first)+
                (p0.second-p1.second)*(p0.second-p1.second));
}

double CalcDis(Point p0,Point p1,Point p2)
{
    return Dis(p0,p1)+Dis(p0,p2)+Dis(p1,p2);
}

void build(int n,double w)
{
    g.clear();
    for (int i = 0;i < n;i++)
        g[make_pair((int)floor(p[i].first/w),(int)floor(p[i].
           second/w))].push_back(p[i]);
}

int main()
{
    int t;
    scanf("%d",&t);
    for (int ft = 1;ft <= t;ft++)
    {
        scanf("%d",&n);
        for (int i = 0;i < n;i++)
        {
            scanf("%lf%lf",&tx,&ty);
            p[i] = make_pair(tx,ty);
        }
        random_shuffle(p,p+n);
        ans = CalcDis(p[0],p[1],p[2]);
        build(3,ans/2.0);
        for (int i = 3;i < n;i++)
        {
            x = (int)floor(2.0*p[i].first/ans);
            y = (int)floor(2.0*p[i].second/ans);
            tmp.clear();
            for (int k = 0;k < 9;k++)
            {
                nx = x+step[k][0];
                ny = y+step[k][1];
                gird = make_pair(nx,ny);
                if (g.find(gird) != g.end())
                {
                    op = g[gird].begin();
                    ed = g[gird].end();
                    for (it = op;it != ed;it++)
                        tmp.push_back(*it);
                }
            }
            flag = false;
            for (int j = 0;j < tmp.size();j++)
                for (int k = j+1;k < tmp.size();k++)
                {
                    nowans = CalcDis(p[i],tmp[j],tmp[k]);
                    if (nowans < ans)
                    {
                        ans = nowans;
                        flag = true;
                    }
                }
            if (flag == true)
                build(i+1,ans/2.0);
            else
                g[make_pair((int)floor(2.0*p[i].first/ans),(int)
                   floor(2.0*p[i].second/ans))].push_back(p[i]);
        }
        printf("%.3f\n",ans);
    }
}
```

7.13 多圆面积并

7.13.1 去重

有时候可能需要去掉不需要的圆

```
for (int i = 0; i < n; i++)
{
    scanf("%lf%lf%lf",&c[i].c.x,&c[i].c.y,&c[i].r);
    del[i] = false;
}
for (int i = 0; i < n; i++)
    if (del[i] == false)
    {
        if (c[i].r == 0.0)  del[i] = true;
        for (int j = 0; j < n; j++)
            if (i != j)
                if (del[j] == false)
                    if (cmp(Point(c[i].c,c[j].c).Len()+c[i].r,c[j
                       ].r) <= 0)
                        del[i] = true;
    }
tn = n;
n = 0;
for (int i = 0; i < tn; i++)
    if (del[i] == false)
        c[n++] = c[i];
```

7.13.2 圆并

$ans[i]$表示被覆盖i次的面积

```
const double pi = acos(-1.0);
const double eps = 1e-8;
struct Point
```

```
{
    double x,y;
    Point(){}
    Point(double _x,double _y)
        {
            x = _x;
            y = _y;
        }
    double Length()
        {
            return sqrt(x*x+y*y);
        }
};
struct Circle
{
    Point c;
    double r;
};
struct Event
{
    double tim;
    int typ;
    Event(){}
    Event(double _tim,int _typ)
        {
            tim = _tim;
            typ = _typ;
        }
};

int cmp(const double& a,const double& b)
{
    if (fabs(a-b) < eps)    return 0;
    if (a < b)  return -1;
    return 1;
}

bool Eventcmp(const Event& a,const Event& b)
{
    return cmp(a.tim,b.tim) < 0;
}

double Area(double theta,double r)
{
    return 0.5*r*r*(theta-sin(theta));
}

double xmult(Point a,Point b)
{
    return a.x*b.y-a.y*b.x;
}

```

```
int n,cur,tote;
Circle c[1000];
double ans[1001],pre[1001],AB,AC,BC,theta,fai,a0,a1;
Event e[4000];
Point lab;

int main()
{
    while (scanf("%d",&n) != EOF)
    {
        for (int i = 0;i < n;i++)
            scanf("%lf%lf%lf",&c[i].c.x,&c[i].c.y,&c[i].r);
        for (int i = 1;i <= n;i++)
            ans[i] = 0.0;
        for (int i = 0;i < n;i++)
        {
            tote = 0;
            e[tote++] = Event(-pi,1);
            e[tote++] = Event(pi,-1);
            for (int j = 0;j < n;j++)
                if (j != i)
                {
                    lab = Point(c[j].c.x-c[i].c.x,c[j].c.y-c[i].c
                        .y);
                    AB = lab.Length();
                    AC = c[i].r;
                    BC = c[j].r;
                    if (cmp(AB+AC,BC) <= 0)
                    {
                        e[tote++] = Event(-pi,1);
                        e[tote++] = Event(pi,-1);
                        continue;
                    }
                    if (cmp(AB+BC,AC) <= 0) continue;
                    if (cmp(AB,AC+BC) > 0)  continue;
                    theta = atan2(lab.y,lab.x);
                    fai = acos((AC*AC+AB*AB-BC*BC)/(2.0*AC*AB));
                    a0 = theta-fai;
                    if (cmp(a0,-pi) < 0)    a0 += 2*pi;
                    a1 = theta+fai;
                    if (cmp(a1,pi) > 0)     a1 -= 2*pi;
                    if (cmp(a0,a1) > 0)
                    {
                        e[tote++] = Event(a0,1);
                        e[tote++] = Event(pi,-1);
                        e[tote++] = Event(-pi,1);
                        e[tote++] = Event(a1,-1);
                    }
                    else
                    {
                        e[tote++] = Event(a0,1);
                        e[tote++] = Event(a1,-1);
                    }
                }
            sort(e,e+tote,Eventcmp);
            cur = 0;
            for (int j = 0;j < tote;j++)
            {
                if (cur != 0 && cmp(e[j].tim,pre[cur]) != 0)
                {
                    ans[cur] += Area(e[j].tim-pre[cur],c[i].r);
                    ans[cur] += xmult(Point(c[i].c.x+c[i].r*cos(pre[cur]),c[i].c.y+c[i].r*sin(pre[cur])),
                                        Point(c[i].c.x+c[i].r*cos(e[j].tim),c[i].c.y+c[i].r*sin(e[j].tim)))/2.0;
                }
                cur += e[j].typ;
                pre[cur] = e[j].tim;
            }
        }
        for (int i = 1;i < n;i++)
            ans[i] -= ans[i+1];
        for (int i = 1;i <= n;i++)
            printf("[%d] = %.3f\n",i,ans[i]);
    }
    return 0;
}
```

7.14 一个圆与多边形面积交

```
bool InCircle(Point a,double r)
{
    return cmp(a.x*a.x+a.y*a.y,r*r) <= 0; //这里判断的时候EPS一定不要太小！！
}

double CalcArea(Point a,Point b,double r)
{
    Point p[4];
    int tot = 0;
    p[tot++] = a;

    Point tv = Point(a,b);
    Line tmp = Line(Point(0,0),Point(tv.y,-tv.x));
    Point near = LineToLine(Line(a,b),tmp);
    if (cmp(near.x*near.x+near.y*near.y,r*r) <= 0)
    {
        double A,B,C;
        A = near.x*near.x+near.y*near.y;
        C = r;
        B = C*C-A;
        double tvl = tv.x*tv.x+tv.y*tv.y;
        double tmp = sqrt(B/tvl); //这样做只用一次开根
        p[tot] = Point(near.x+tmp*tv.x,near.y+tmp*tv.y);
        if (OnSeg(Line(a,b),p[tot]) == true)    tot++;
        p[tot] = Point(near.x-tmp*tv.x,near.y-tmp*tv.y);
        if (OnSeg(Line(a,b),p[tot]) == true)    tot++;
    }
    if (tot == 3)
    {
        if (cmp(Point(p[0],p[1]).Length(),Point(p[0],p[2]).Length
           ()) > 0)
            swap(p[1],p[2]);
    }
    p[tot++] = b;

    double res = 0.0,theta,a0,a1,sgn;
    for (int i = 0;i < tot-1;i++)
    {
        if (InCircle(p[i],r) == true && InCircle(p[i+1],r) ==
           true)
        {
            res += 0.5*xmult(p[i],p[i+1]);
        }
        else
        {
            a0 = atan2(p[i+1].y,p[i+1].x);
            a1 = atan2(p[i].y,p[i].x);
            if (a0 < a1)    a0 += 2*pi;
            theta = a0-a1;
            if (cmp(theta,pi) >= 0) theta = 2*pi-theta;
            sgn = xmult(p[i],p[i+1])/2.0;
            if (cmp(sgn,0) < 0) theta = -theta;
            res += 0.5*r*r*theta;
        }
    }
    return res;
}
```

调用

```
area2 = 0.0;
for (int i = 0;i < resn;i++) //遍历每条边，按照逆时针
    area2 += CalcArea(p[i],p[(i+1)%resn],r);
```

7.15 精度问题

7.15.1 浮点数为啥会有精度问题

浮点数(以C/C++为准)，一般用的较多的是float、double。

	占字节数	数值范围	十进制精度位数
float	4	$-3.4e-38 \sim 3.4e38$	$6 \sim 7$
double	8	$-1.7e-308 \sim 1.7e308$	$14 \sim 15$

如果内存不是很紧张或者精度要求不是很低，一般选用double。14位的精度(是有效数字位，

不是小数点后的位数)通常够用了。注意，问题来了，数据精度位数达到了14位，但有些浮点运算的结果精度并达不到这么高，可能准确的结果只有$10 \sim 12$位左右。那低几位呢？自然就是不可预料的数字了。这给我们带来这样的问题：即使是理论上相同的值，由于是经过不同的运算过程得到的，他们在低几位有可能(一般来说都是)是不同的。这种现象看似没太大的影响，却会一种运算产生致命的影响：==。恩，就是判断相等。注意，C/C++中浮点数的==需要完全一样才能返回true。

7.15.2 eps

eps缩写自epsilon，表示一个小量，但这个小量又要确保远大于浮点运算结果的不确定量。eps最常见的取值是$1e-8$左右。引入eps后，我们判断两浮点数a、b相等的方式如下：

```
int sgn(double a){return a < -eps ? -1 : a < eps ? 0 : 1;}
```

这样，我们才能把相差非常近的浮点数判为相等；同时把确实相差较大(差值大于eps)的数判为不相等。
养成好习惯，尽量不要再对浮点数做==判断。

7.15.3 eps带来的函数越界

如果sqrt(a), asin(a), acos(a) 中的a是你自己算出来并传进来的，那就得小心了。
如果a本来应该是0的，由于浮点误差，可能实际是一个绝对值很小的负数（比如$-1e-12$），这样sqrt(a)应得0的，直接因a不在定义域而出错。
类似地，如果a本来应该是±1,则asin(a)、acos(a)也有可能出错。
因此，对于此种函数，必需事先对a进行校正。

7.15.4 输出陷阱I

现在考虑一种情况，题目要求输出保留两位小数。有个case的正确答案的精确值是0.005,按理应该输出0.01，但你的结果可能是0.005000000001(恭喜)，也有可能是0.004999999999(悲剧)，如果按照printf("%.2lf", a)输出，那你的遭遇将和括号里的字相同。
解决办法是，如果a为正，则输出$a+eps$, 否则输出$a-eps$

7.15.5 输出陷阱II

ICPC题目输出有个不成文的规定(有时也成文)，不要输出：-0.000
那我们首先要弄清，什么时候按printf("%.3lf", a)输出会出现这个结果。
直接给出结果好了：$a \in (-0.000499999\cdots, -0.000\cdots1)$
所以，如果你发现a落在这个范围内，请直接输出0.000。更保险的做法是用sprintf直接判断输出结果是不是-0.000再予处理。

7.15.6 范围越界

请注意，虽然double可以表示的数的范围很大，却不是不穷大，上面说过最大是$1e308$。所以有些时候你得小心了，比如做连乘的时候，必要的时候要换成对数的和。

7.15.7 关于set

经观察,set不是通过==来判断相等的，是通过<来进行的，具体说来，只要$a < b$ 和$b < a$都不成立，就认为a和b相等，可以发现，如果将小于定义成:

```
bool operator < (const Dat dat)const{return val < dat.val - eps;}
```

就可以解决问题了。(基本类型不能重载运算符，所以封装了下)

7.15.8 输入值波动过大

这种情况不常见，不过可以帮助你更熟悉eps。假如一道题输入说，给一个浮点数a, $1e-20 < a < 1e20$。那你还敢用$1e-8$做eps么？合理的做法是把eps按照输入规模缩放到合适大小。

7.15.9 一些建议

容易产生较大浮点误差的函数有asin、 acos。欢迎尽量使用atan2。
另外，如果数据明确说明是整数，而且范围不大的话，使用int或者long long代替double都是极佳选择，因为就不存在浮点误差了

8 搜索

8.1 Dancing Links

仰慕罗神。

```
void remove1(int col)
{
    int i,j;
    L[R[col]]=L[col];
    R[L[col]]=R[col];
    for(i=D[col];i!=col;i=D[i])
    {
        L[R[i]]=L[i];
        R[L[i]]=R[i];
    }
}
void remove2(int col)
{
    int i,j;
    L[R[col]]=L[col];
    R[L[col]]=R[col];
    for(i=D[col];i!=col;i=D[i])
    {
        for(j=R[i];j!=i;j=R[j])
        {
            U[D[j]]=U[j];
            D[U[j]]=D[j];
            --nk[C[j]];
        }
    }
}
void resume1(int col)
{
    int i,j;
    for(i=U[col];i!=col;i=U[i])
    {
        L[R[i]]=i;
        R[L[i]]=i;
    }
    L[R[col]]=col;
    R[L[col]]=col;
}
void resume2(int col)
{
    int i,j;
    for(i=U[col];i!=col;i=U[i])
    {
        for(j=L[i];j!=i;j=L[j])
        {
            ++nk[C[j]];
            U[D[j]]=j;
            D[U[j]]=j;
        }
    }
    L[R[col]]=col;
    R[L[col]]=col;
}
int h()
{
  bool vis[100];
  memset(vis,false,sizeof(vis));
  int i,j,k,res=0,mi,col;
  while(1)
  {
    mi=inf;
    for(i=R[head];i!=head&&i<=2*n;i=R[i])
      if(mi>nk[i]&&!vis[i])
      {
        mi=nk[i];
        col=i;
      }
    if(mi==inf)
      break;
    res++;vis[col]=true;
    for(j=D[col];j!=col;j=D[j])
      for(k=R[j];k!=j;k=R[k])
      {
        if(C[k]>2*n)
          continue;
        vis[C[k]]=true;
      }
  }
  return res;
}
bool DLX(int d,int deep)
{
  if(d+h()>deep) return false;
    if(R[head]==head||R[head]>2*n)
      return true;
    if(d>=deep)
      return false;
    int col,ma=inf;
    int i,j;
    for(i=R[head];i!=head&&i<=2*n;i=R[i])
        if(nk[i]<ma)
        {
            col=i;
            ma=nk[i];
        }
    remove1(col);
    for(i=D[col];i!=col;i=D[i])
    {
        int flag=1;
        for(j=R[i];;j=R[j])
        {
            if(j==R[i]&&!flag)
                break;
            U[D[j]]=U[j];
            D[U[j]]=D[j];
            if(C[j]>2*n)
                remove2(C[j]);
            else
                remove1(C[j]);
            flag=0;
        }
        if(DLX(d+1,deep))
          return true;
        flag=1;
        for(j=L[i];;j=L[j])
        {
            if(j==L[i]&&!flag)
                break;
            if(C[j]>2*n)
                resume2(C[j]);
            else
                resume1(C[j]);
            U[D[j]]=j;
            D[U[j]]=j;
            flag=0;
        }
    }
    resume1(col);
    return false;
}
```

9 杂物

9.1 高精度数

支持乘以整数和加法。

```
struct BigInt
{
    const static int mod = 100000000;
    int a[600],len;
    BigInt (){}
    BigInt (int v)
    {
        len = 0;
        do
        {
            a[len++] = v%mod;
            v /= mod;
        }while(v);
    }
    BigInt operator *(const int& b) const
    {
        BigInt res;
        res.len = len;
        for (int i = 0; i <= len; ++i)
            res.a[i] = 0;
        for (int i = 0; i < len; ++i)
        {
            res.a[i] += a[i]*b;
            res.a[i+1] += res.a[i]/mod;
            res.a[i] %= mod;
        }
        if (res.a[len] > 0)  res.len++;
        return res;
    }
    BigInt operator +(const BigInt& b) const
    {
        BigInt res;
        res.len = max(len,b.len);
        for (int i = 0; i <= res.len; ++i)
            res.a[i] = 0;
        for (int i = 0; i < res.len; ++i)
        {
            res.a[i] += ((i < len)?a[i]:0)+((i < b.len)?b.a[i]:0)
               ;
            res.a[i+1] += res.a[i]/mod;
            res.a[i] %= mod;
        }
        if (res.a[res.len] > 0)  res.len++;
        return res;
    }
    void output()
    {
        printf("%d",a[len-1]);
        for (int i = len-2; i >= 0; --i)
            printf("%08d",a[i]);
        printf("\n");
    }
};
```

9.2 整数外挂

```
int wg;
char ch;
bool ng;

inline int readint()
{
    ch = getchar();
    while (ch != '-' && (ch < '0' || ch > '9')) ch = getchar();
    if (ch == '-')
    {
        ng = true;
        ch = getchar();
    }
    else
        ng = false;
    wg = ch-'0';
    ch = getchar();
    while (ch >= '0' && ch <= '9')
    {
        wg = wg*10+ch-'0';
        ch = getchar();
    }
    if (ng == true) wg = -wg;
    return wg;
}
```

9.3 Java

9.3.1 IO

9.3.2 优先队列

```
PriorityQueue queue = new PriorityQueue( 1, new Comparator()
{
    public int compare( Point a, Point b )
    {
  if( a.x < b.x || a.x == b.x && a.y < b.y )
      return -1;
  else if( a.x == b.x && a.y == b.y )
      return 0;
  else
      return 1;
    }
});
```

9.3.3 Map

```
Map map = new HashMap();
map.put("sa","dd");
String str = map.get("sa").toString;

for(Object obj : map.keySet()){
    Object value = map.get(obj );
}
```

9.3.4 sort

```
static class cmp implements Comparator
{
    public int compare(Object o1,Object o2)
    {
  BigInteger b1=(BigInteger)o1;
  BigInteger b2=(BigInteger)o2;
  return b1.compareTo(b2);
    }
}
public static void main(String[] args) throws IOException
{
    Scanner cin = new Scanner(System.in);
    int n;
    n=cin.nextInt();
    BigInteger[] seg = new BigInteger[n];
    for (int i=0;i<n;i++)
  seg[i]=cin.nextBigInteger();
    Arrays.sort(seg,new cmp());
}
```

9.4 hashmap

```
struct hash_map
{
    const static int mod=10007;
    int head[mod];
    struct hash_tables
    {
        int key;
        int val;
        int next;
    } ele[10007];
    int N;
    int getHash(int x)
    {
        return x%mod;
    }
    void init()
    {
        memset(head,255,sizeof(head));
```

```
            N=0;
        }
        void clear()
        {
            for (int i = 0; i < N; i++)
                head[getHash(ele[i].key)] = -1;
            N = 0;
        }
        int fint(int x)
        {
            for (int i=head[getHash(x)]; i!=-1; i=ele[i].next)
                if (ele[i].key==x) return i;
            return -1;
        }
        void insert(int x)
        {
            int tmp=getHash(x);
            ele[N].key=x;
            ele[N].val=0;
            ele[N].next=head[tmp];
            head[tmp]=N++;
        }
        int& operator [](int x)
        {
            int tmp=fint(x);
            if (tmp==-1)
            {
                insert(x);
                return ele[N-1].val;
            }
            else
                return ele[tmp].val;
        }
};
```

9.5 C++&STL常用函数

9.5.1 lower_bound/upper_bound

不解释

```
iterator lower_bound(const key_type &key )\\返回一个迭代器，指向键值>=
    key的第一个元素。
iterator upper_bound(const key_type &key )\\返回一个迭代器，指向键值>
    key的第一个元素。

#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;

int main () {
  int myints[] = {10,20,30,30,20,10,10,20};
  vector<int> v(myints,myints+8);           // 10 20 30 30 20 10 10 20
  vector<int>::iterator low,up;

  sort (v.begin(), v.end());                // 10 10 10 20 20 20 30 30

  low=lower_bound (v.begin(), v.end(), 20); //          ^
  up= upper_bound (v.begin(), v.end(), 20); //                   ^

  cout << "lower_bound␣at␣position␣" << int(low- v.begin()) << endl;
  cout << "upper_bound␣at␣position␣" << int(up - v.begin()) << endl;

  return 0;
}
```

Output:

```
lower_bound at position 3
upper_bound at position 6
```

9.5.2 rotate

把数组后一半搬到前面

```
template <class ForwardIterator>
  void rotate ( ForwardIterator first, ForwardIterator middle,
                ForwardIterator last );
```

9.5.3 nth_element

```
template <class RandomAccessIterator>
  void nth_element ( RandomAccessIterator first, RandomAccessIterator nth,
                     RandomAccessIterator last );

template <class RandomAccessIterator, class Comapre>
  void nth_element ( RandomAccessIterator first, RandomAccessIterator nth,
                     RandomAccessIterator last, Compare comp );
```

9.5.4 bitset

取用

```
bitset<4> mybits;

mybits[1]=1;             // 0010
mybits[2]=mybits[1];     // 0110
```

翻转

```
bitset<4> mybits (string("0001"));

cout << mybits.flip(2) << endl;     // 0101
cout << mybits.flip() << endl;      // 1010
```

运算

```
bitset<4> first (string("1001"));
bitset<4> second (string("0011"));

cout << (first^=second) << endl;          // 1010 (XOR,assign)
cout << (first&=second) << endl;          // 0010 (AND,assign)
cout << (first|=second) << endl;          // 0011 (OR,assign)

cout << (first<<=2) << endl;              // 1100 (SHL,assign)
cout << (first>>=1) << endl;              // 0110 (SHR,assign)

cout << (~second) << endl;                // 1100 (NOT)
cout << (second<<1) << endl;              // 0110 (SHL)
cout << (second>>1) << endl;              // 0001 (SHR)

cout << (first==second) << endl;          // false (0110==0011)
cout << (first!=second) << endl;          // true  (0110!=0011)

cout << (first&second) << endl;           // 0010
cout << (first|second) << endl;           // 0111
cout << (first^second) << endl;           // 0101
```

9.5.5 multimap

遍历

```
multimap<char,int> mymm;
multimap<char,int>::iterator it;
char c;

mymm.insert(pair<char,int>('x',50));
mymm.insert(pair<char,int>('y',100));
mymm.insert(pair<char,int>('y',150));
mymm.insert(pair<char,int>('y',200));
mymm.insert(pair<char,int>('z',250));
mymm.insert(pair<char,int>('z',300));

for (c='x'; c<='z'; c++)
{
  cout << "There␣are␣" << (int)mymm.count(c);
  cout << "␣elements␣with␣key␣" << c << ":";
  for (it=mymm.equal_range(c).first; it!=mymm.equal_range(c).
     second; ++it)
    cout << "␣" << (*it).second;
  cout << endl;
}
/*
Output:

There are 1 elements with key x: 50
There are 3 elements with key y: 100 150 200
There are 2 elements with key z: 250 300
*/
```

二分查找

```
multimap<char,int> mymultimap;
multimap<char,int>::iterator it,itlow,itup;

mymultimap.insert(pair<char,int>('a',10));
mymultimap.insert(pair<char,int>('b',121));
mymultimap.insert(pair<char,int>('c',1001));
mymultimap.insert(pair<char,int>('c',2002));
mymultimap.insert(pair<char,int>('d',11011));
mymultimap.insert(pair<char,int>('e',44));

itlow=mymultimap.lower_bound ('b');  // itlow points to b
itup=mymultimap.upper_bound ('d');   // itup points to e (not d)

// print range [itlow,itup):
for ( it=itlow ; it != itup; it++ )
  cout << (*it).first << "␣=>␣" << (*it).second << endl;

/*
Output:

b => 121
c => 1001
c => 2002
d => 11011
*/
```

删除

```
multimap<char,int> mymultimap;
multimap<char,int>::iterator it;

// insert some values:
mymultimap.insert(pair<char,int>('a',10));
mymultimap.insert(pair<char,int>('b',20));
mymultimap.insert(pair<char,int>('b',30));
mymultimap.insert(pair<char,int>('c',40));
mymultimap.insert(pair<char,int>('d',50));
mymultimap.insert(pair<char,int>('d',60));
mymultimap.insert(pair<char,int>('e',70));
mymultimap.insert(pair<char,int>('f',80));

```

```
it=mymultimap.find('b');
mymultimap.erase (it);                    // erasing by iterator
    (1 element)

mymultimap.erase ('d');                   // erasing by key (2
   elements)

it=mymultimap.find ('e');
mymultimap.erase ( it, mymultimap.end() ); // erasing by range

// show content:
for ( it=mymultimap.begin() ; it != mymultimap.end(); it++ )
  cout << (*it).first << " => " << (*it).second << endl;

/*
Output:

a => 10
b => 30
c => 40
*/
```

9.6 其它

9.6.1 对跑脚本

```
while true; do
  ./gen > input
  ./sol < input > output.sol
  ./bf < input > output.bf

  diff output.sol output.bf
  if [ $? -ne 0 ] ; then break; fi
done
```

9.6.2 枚举长为n含k个1的01串

```
int n = 5,k = 3;
for (int s = (1 << k)-1,u = 1 << n; s < u;)
{
    for (int i = 0;i < n;i++)
        printf("%d",(((s>>(n-1-i))&1) == 1));
    printf("\n");

    int b = s & -s;
    s = (s+b)|(((s^(s+b))>>2)/b);
}
```